# DISTRIBUTED DATA RECONCILIATION AND BIAS ESTIMATION WITH NON-GAUSSIAN NOISE FOR SENSOR NETWORK

JOE YEN YEN

(*B.Eng.(Hons), M.Eng., NUS*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL

FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2010

# Acknowledgment

Advisors always come first in the acknowledgment, and without aiming for political correctness, I would say it is rightly so. As much as the PhD has been a hard time for me, I believe it has not been any less so for my advisors. I am grateful to Dr Lim Khiang Wee, Prof Ling Keck Voon, Prof Ho Weng Khuen and Dr Zhang Jing Bing, without whom this thesis would never have been. Prof Ling Keck Voon and Prof Ho Weng Khuen, in particular, have devoted immeasurable time and effort to pull this thesis into being.

The generous help and guidance of Prof Jose Alberto Romagnoli of Louisiana State University during the initial conception of the thesis is also greatly acknowledged.

During the long PhD journey, many precious individuals have been my pillars of support. My friends Zhao Sumin, Quek Boon Kiat, Lee Hui Mien, Kiew Choon Meng, Syahfitri Undaya, Sri Winarsih, Valdomiro Peixoto, and many others, have supported me in more ways than they know. My family has been the force that sustains my journey. My in-laws have provided family comfort away from home.

Finally, my husband, Ng Boon Ping, has shared everything with me through times of darkness. No words are significant enough to express my gratitude.

# Table of Contents

# Summary

The advancement of sensor network technology presents both a new platform and a challenging environment for sensing applications. An important challenge is to incorporate techniques to remove measurement corruptions, to which sensors are perpetually prone. Data reconciliation (DR) is a measurement adjustment technique commonly used in the process industry to deal with measurement corruptions. It improves measurement accuracy by ensuring their consistency; measurements are adjusted according to known relationships among the measured variables and based on the statistical characteristics of the sensor precision. However, DR has traditionally been performed in a centralized manner, where measurements are collected from all sensors to a central node to be processed. This thesis considers DR in a distributed sensor network environment and distributes the linear steady-state DR computation to the nodes in the sensor network. The distributed DR (DDR) is derived, and an implementation algorithm is developed. As each sensor node actively participates in the distributed DR, it is robust to the failure of any node, and gracefully degrades when more than one nodes fail. Illustrative examples are presented to demonstrate the proposed DDR, while an application case study of an experimental-scale chemical

plant demonstrates its usefulness.

Sensor biases are prevalent in all sensing applications, and bias estimation proves an important tool in ensuring measurement accuracy. In this thesis, instead of collecting all measurements to a central processing node to estimate their biases, the intelligence of the sensor nodes is leveraged to perform bias estimation in a distributed manner. The performance of the Generalized T (GT), inter-quartile range test cum least-square (IQR+LS) and least-square (LS) bias estimators used in the distributed bias estimation (DBE) are analyzed through both theoretical tools and experiments. The theoretical tools relate the estimator type and sample size with the estimation variance. As such, besides providing a basis for theoretical performance comparison among the bias estimators, the theoretical tools allow one to design the estimator to achieve specified variance, or provide one with an expected estimator precision for a given set of estimator parameters and sample size. The theoretical results are verified experimentally with the application case study of an experimental-scale chemical plant.

# List of Tables

# List of Figures

# Chapter 1. Introduction

## 1.1 Motivation

Intelligent sensor network is a collection of autonomous devices that measure characteristics of their environment, perform local computations, and communicate with one another over a network [1]. Termed sensor node, each of these devices is typically small, low cost and battery operated [2]. These characteristics make them very attractive: their compact size means that they can be placed inconspicuously without disrupting the environment that they are sensing [3]; the low cost means that they can be deployed in large number, resulting in dense deployment with high redundancy; and being wireless and battery-operated, they are free from the constraints of communication and power infrastructure, such that they can be placed anywhere and their placement can be easily adjusted, for example to optimize coverage of the phenomena or detectability of an event.

Sensor measurements, however, are prone to corruptions. As inexpensive sensors are used in sensor networks to achieve dense deployment and perhaps, the required minimal form factor, they tend to be corrupted or fail more easily. Techniques to remove these measurement corruptions are therefore especially important in sensor

networks applications [1,3,21].

Data reconciliation (DR) [4] is a measurement adjustment technique commonly used in the process industry to deal with measurement corruptions. It improves measurement accuracy by ensuring their consistency; measurements are adjusted according to known relationships among the measured variables and based on the statistical characteristics of the sensor precision. The sensor network deployment makes it suitable for application of DR, as the sensors usually measure spatially correlated signals. Such correlations mean that there exist functional relationships describing the behaviors of the measured signals in terms of one another, which are therefore fitting for use as a basis for reconciliation of the sensor measurements. In fact, reconciliation of measurements is the procedure that enables the physical redundancy of the sensor deployment to be leveraged, to compensate for the lower quality of the low-cost sensors.

A straightforward way to perform DR in sensor networks is to download the measurements from all sensor nodes in the network, and then have a central processing node carry out the reconciliation [1]. In this case, each sensor node need not possess any knowledge of correlations with other nodes, nor perform any computation, nor engage in meaningful communication (collaboration) with other nodes. The central node must therefore maintain the knowledge of correlations among all sensors, perform all necessary algorithmic steps and handle transmissions to and from all the sensor nodes. Although there are cases in which the centralized approach is more appropriate, for example, when it is desired to collect all the raw measurements to a central location, there are good reasons to prefer a distributed approach [1,5].

A major drawback of the centralized approach is that there is a communication and

computation bottleneck at the central processing node [5]. A crucial implication of this is that the DR processing is critically dependent on the availability of the central processing node. Any disruption or failure of the central processing node will affect or worse, halt, the DR processing. There is also a gross under-utilization of the computational power of the intelligent nodes in the network. A scheme that leverages the capabilities of the intelligent sensors to eliminate dependence on a central processing node is therefore desired.

While DR reduces the effect of random noise on data, assumptions made by conventional DR approaches are often restrictive. More specifically, the conventional least square estimator has an implicit assumption on the normality of data. However, data in practice are more often than not subjected to the occurrence of outliers, transients in a supposedly steady-state period, instrument failure, human error and other process nature that renders the data non-normal. If approaches based on normality assumption are used on the non-normal data, poor estimates may result. For example, a single huge outlier can skew the least-square estimates significantly. It is therefore imperative to consider data processing approaches that are robust to outliers [11, 12].

As DR focuses on the treatment of random measurement corruptions, additional steps must be taken to correct for systematic measurement corruptions. Bias, which can be caused by miscalibration of sensors or some instrument malfunction, is a prevalent type of systematic corruption [17]. The intelligent sensor nodes must therefore be equipped with capabilities to treat bias [3,6-7,22-26].

In this thesis, a strategy to deal with outliers and biases is proposed. The Generalized T (GT) distribution is chosen to represent the measurement noise. With this approach, there is no assumption of normality (Gaussian distribution) on the data,

and outliers are modeled instead of removed from the data set. Furthermore, as a more general distribution, the GT can adapt to many common distributions, including the Gaussian distribution, through varying the GT distribution parameters. The proposed GT-based strategy also makes use of linear consistency model relating measurements of neighbouring sensors, hence utilising the spatial redundancy among the sensors.

Two relevant topics in sensor network are known as online sensor data cleaning and distributed calibration. In the following, several representative works under these topics are described and compared with the work in this thesis.

In contrast to the proposed GT-based strategy, other popular approaches in the field of sensor network [21,27,30], as summarized below, assume Gaussian distribution of the measurement data. Using these approaches, outliers are detected/ identified through statistical tests based on Gaussian assumption, before being removed from the data.

The work of Elnahrawy and Nath [21] seems to be exemplary in online sensor data cleaning in sensor network. In this work, Bayesian estimation is used to give more accurate estimate of the true value measured by a sensor, given the measurement of the sensor, random characteristic of the measurement (likelihood) and the prior distribution of the true measured value. Cleaning is therefore done in a single sensor basis, without making use of related measurements from a node's neighbours (spatial redundancy).

Spatial information is taken into consideration in the more recent work of Ji and Szczodrak [27], where some estimate of covariances among neighbouring nodes are obtained using steady-state data. The $\chi^2$ test based on the estimated covariances is then performed on tuples containing the measurements of the neighbouring nodes, to detect and identify outliers. The use of the $\chi^2$ test implies Gaussian assumption on

the measurement noise.

Jeffery et al [28] proposed a multi-tiered architectural framework to clean sensor data. Both temporal and spatial redundancies are considered; however, as the main focus of the work is to propose the architectural framework, only very simple, heuristic outlier detection/identification and smoothing (replacement of outliers with interpolated values) techniques are presented in the form of declarative queries. However, similarly heuristic techniques to deal with outliers seem to be common in other online sensor cleaning approaches [29]. For example, Mukhopadhyay et al [29] uses a tree-structure decision analysis coupled with ARMA prediction model to estimate a "true value" for a sensor measurement. Decision is then made to either use the actual measurement of the sensor or the estimated value as the corrected/cleaned data point.

The use of linear consistency model has been considered under the topic of calibration in sensor network, i.e. in the work of Balzano et al [6]. However, although the importance of distributed implementation has been emphasized, this work does not outline the distributed algorithm of the proposed calibration method.

## 1.2 Contributions

The distributed data reconciliation (DDR) is derived to enable a group of intelligent sensors to perform DR in-network in a distributed manner. Algebraic analysis of the conventional (centralized) DR is conducted and the distributed DR is formulated.

An implementation algorithm for the DDR is developed. In the proposed DDR algorithm, each sensor node is made aware of itself and its neighbours, is fully in charge of computations and coordination with other nodes to reconcile its own data,

and is able to respond to abnormal situation such as a missing/failed neighbouring node. The dependence on a central processing node to perform DR is eliminated, making the distributed DR robust to the failure of the central processing node.

A case study is conducted by applying DDR in an experimental-scale chemical plant to demonstrate the procedures of DDR and its usefulness in maintaining operation despite node failures.

To handle biases and outliers in the sensor measurements, the distributed bias estimation (DBE) with the Generalized T (GT) estimator is derived and its implementation algorithm developed. Similar to DDR, DBE enables a group of intelligent sensors to perform bias estimation (BE) in-network in a distributed manner, such that the dependence on a central processing node to perform bias estimation is eliminated.

For comparative studies with the GT-based DBE, the inter-quartile range test cum least square (IQR+LS) and the traditional least-square (LS) estimators are also applied in DBE. The performance of these estimators are analyzed using theoretical tools based on the Influence Function (IF). In the light of the equations derived in this thesis, an efficient estimator can be selected. In the presence of outliers that are close to good data, the equations show that using GT, instead of normal distribution, to characterize sensor data gives rise to a more efficient estimator than the LS and IQR+LS in terms of estimation variance.

The case study of the experimental-scale chemical plant is also conducted to demonstrate the procedures of DBE and to study the performance of the GT, IQR+LS and LS estimators.

.

## 1.3 Outline of the thesis

This thesis is organized as follows. Chapter 2 presents the proposed distributed data reconciliation (DDR). The proposed DDR is applied to a case study of an experimental-scale chemical plant in Chapter 3. Chapter 4 presents the proposed distributed bias estimation (DBE). The case study of the experimental-scale chemical plant is also conducted for the DBE, with the experiment described and the results discussed in Chapter 5. Finally, Chapter 6 presents the conclusion of the thesis.

# Chapter 2. Distributed Data Reconciliation (DDR)

## 2.1 Introduction

In this chapter, distributed DR is derived and an implementation algorithm is given. The goal is for DR to be performed entirely in-network, hence eliminating problems associated with having a central processing node. Indeed, in the proposed approach, each sensor node actively participates and takes responsibility in reconciling their own and their neighbors' measurements, through local computation and collaboration. In this case, not only are the computation and communication capabilities of the sensor nodes in the network leveraged, but also, a certain level of autonomy, or at least awareness, is assigned to the sensor nodes. With such autonomy/awareness, these nodes can collaborate and reconcile with their neighbouring nodes, hence reducing unnecessary communication with other parts of the network. In addition, a certain level of parallelism can be achieved across groups of independent nodes.

By distributing the processing and communication burden in a meaningful way to each sensor node, the distributed DR is more robust to failures of one or more nodes.

In contrast, the centralized approach is vulnerable to the failure of the central processing node, which will result in the complete failure of DR processing.

In the field of sensor network, the topic of reconciling sensor measurements using consistency model has recently gained attention. Termed model-based distributed sensor calibration [6], this topic is probably best represented by the works of Bychkovskiy et al [7] and Guestrin et al [1]. The main difference between the work in this chapter and these two works is in the consistency model used. Bychkovskiy et al [7] assumed very dense deployment, such that all sensors measure the same variable. Guestrin et al [1], on the other hand, worked in the probabilistic inference domain and as such, their consistency model is in the form of joint probability distribution of the measured variables. The work in this chapter, however, uses any linear consistency model and set the reconciliation problem as an optimization problem using the linear model as constraints. Balzano et al [6] also proposed a strategy for model-based sensor calibration that uses a range of linear models. In relation to [6], the work in this chapter may be seen as an extension in terms of providing a distributed framework for linear model-based sensor calibration.

This chapter is organized as follows. In the following section, the topic of DR is first introduced. The proposed DDR is then presented in detail in Section 2.3. Section 2.4 ends the chapter with concluding remarks.

## 2.2 Data Reconciliation (DR)

Data reconciliation is well-studied in the area of process engineering [4,17-20] and only the relevant equations necessary for the derivation of the distributed algorithm are given. The mathematical formulation of the linear steady-state DR is as follows.

Let there be $n$ sensors, making $y = [y_1 \dots y_n]^T$ measurements.

*Measurement model:* In the absence of gross errors, the measurement model can be expressed as:

$$y = x + \varepsilon, \tag{2.1}$$

where $x = [x_1 \dots x_n]^T$ are the actual measured variable and $\varepsilon$ represent the measurement noises of each sensor for the particular set of measurements. Furthermore, let the measurement noise have the following characteristics:

The expected value of $\varepsilon$ is a null vector, i.e.

$$E\{\varepsilon\} = 0. \tag{2.2}$$

The covariance matrix of the measurement errors is known and positive definite, i.e.

$$Cov(\varepsilon) = \Psi = E\{\varepsilon\varepsilon^T\}. \tag{2.3}$$

*Consistency relationship model, or DR constraints:* The DR constraints consist of equations describing how the measured variables $x = [x_1 \dots x_n]^T$ are inter-related. In linear steady-state DR, it can be expressed in the matrix form:

$$Ax = 0, \tag{2.4}$$

where each row of $A$ is one constraint, i.e. a linear equation relating $x_1 \ldots x_n$.

*The DR problem:* With the measurement model in (2.1), assumptions on the noise in (2.2)-(2.3), and the constraints in (2.4), the DR is cast as a constrained weighted least-square problem:

$$\begin{aligned} &\min_{\varepsilon} \varepsilon^T \Psi^{-1} \varepsilon \\ &\text{s.t. } A(y-\varepsilon) = 0 \end{aligned} . \tag{2.5}$$

*Solution of the DR problem:* Using Lagrange multiplier approach, the DR problem above can be solved for the estimates:

error estimate:

$$\hat{\varepsilon} = \Psi A^T (A \Psi A^T)^{-1} A y \tag{2.6}$$

reconciled estimate:

$$\hat{x} = y - \hat{\varepsilon} \tag{2.7}$$

estimate covariance:

$$\text{cov}(\hat{x}) = \hat{\Psi} = \Psi - \Psi A^T (A \Psi A^T)^{-1} A \Psi . \tag{2.8}$$

Note that for a unique solution to exist, the constraint matrix $A$ must satisfy the usual conditions for solvability of the least square problem, i.e. if $A$ is an $m \times N$ matrix, then $m < N$ and $rank(A) = m$.

## 2.3 Distributed Data Reconciliation (DDR)

### 2.3.1 Overview

This section presents in detail the proposed distributed data reconciliation (DDR) algorithm. Due to the rather involved algorithms of the distributed DR, illustrative examples of a two-node (Section 2.3.2) and three-node (Section 2.3.3) networks are provided in this section to give the reader a basic understanding of the distributed DR algorithm and to contrast it with the centralized DR. Section 2.3.3 then describes the general distributed DR for an *N*-node network, while the detailed implementation algorithm can be found in Appendix 2A.

### 2.3.2 Example 1: A two-node network

Consider a sensor network consisting of two sensor nodes, related through a constraint $x_1 - x_2 = 0$ (Figure 2.1). Given the measurements of the two sensors:

$$y = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T = \begin{bmatrix} 10+\sqrt{2} & 8 \end{bmatrix}^T,$$

with their corresponding variances:

$$\sigma_1^2 = 1,\ \sigma_2^2 = 2.$$

The DR problem can be stated as (2.5), where in this case,

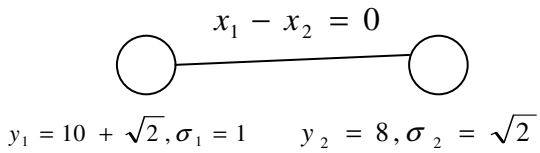



Figure 2.1. A two-node sensor network

$$\varepsilon = [\varepsilon_1 \quad \varepsilon_2]^T = [y_1 - x_1 \quad y_2 - x_2]^T,$$

$$A = [a_{11} \quad a_{12}] = [1 \quad -1],$$

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} \\ \psi_{21} & \psi_{22} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \\ & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 1 & \\ & 2 \end{bmatrix}.$$

## Centralized DR:

In a centralized scheme, all processing is done at a single location. Assume that in this example, the single location is at Node 2. The reconciled estimates can be calculated by Node 2 in a centralized manner using (2.6)-(2.8), resulting in

$$\hat{\varepsilon} = [1 \quad -2]^T,$$

$$\hat{x} = \left[9\tfrac{1}{3} + \tfrac{2}{3}\sqrt{2} \quad 9\tfrac{1}{3} + \tfrac{2}{3}\sqrt{2}\right]^T,$$

and

$$\hat{\Psi} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}.$$

Obviously, in order to carry out the above calculations, Node 2 must have the values of $y_1$, $y_2$, $\sigma_1$, $\sigma_2$ and *A*. The measurement $y_1$ is usually an average of readings taken by the sensor in Node 1, while $\sigma_1$ is the variance of the readings. This means Node 1 has to send a number of its sensor readings to Node 2, so that Node 2 can compute $y_1$ and $\sigma_1$. Also, the constraint matrix *A* must be stored in Node 2.

## Distributed DR:

In the proposed distributed DR, instead of Node 2 doing all the processing, Node 1 and Node 2 share the processing and communicate to complete the data reconciliation. Table 2.1 (which can be found at the end of this chapter) shows the details. Node 1 and 2 each computes and holds its own measurement average and variance, and keeps its own constraint and covariances relating it with each other, as seen in the initialization step, Step 0 of Table 2.1.

To compute the reconciled estimates and estimate covariances, the nodes need data from each other, in addition to their own local data. To do this, the nodes go through a series of computation and communication procedures as follows (see also Steps 1-3 of Table 2.1):

*(i) Computation of local results using local node data:*

Node 1 computes, locally:

$$r_1 = a_{11} y_1 = 10 + \sqrt{2},$$
$$\theta_1 = a_{11}\psi_{11} + a_{12}\psi_{12} = a_{11}\psi_{11} = 1 \ \left(\text{since } \psi_{12} = 0 \text{ at Node 1}\right).$$

Node 2 computes, locally:

$$r_2 = a_{12} y_2 = -8,$$
$$\theta_2 = a_{11}\psi_{21} + a_{12}\psi_{22} = a_{12}\psi_{22} = -2 \ \ (\text{since } \psi_{21} = 0 \text{ at Node 2}).$$

*(ii) Aggregation of local results to compute reconciled estimates and covariances:*

After local computation, Node 1 sends its local results $(r_1, \theta_1)$ to Node 2, which then aggregates them with its local results as follows:

$$r = r_1 + r_2 = 2 + \sqrt{2},$$
$$\phi = a_{11}\theta_1 + a_{12}\theta_2 = 3.$$

With these aggregate values, Node 2 can now compute its own reconciled estimate and covariances, as follows:

$$\hat{x}_2 = y_2 - \hat{\varepsilon}_2 = 9\tfrac{1}{3} + \tfrac{2}{3}\sqrt{2}; \text{ where } \hat{\varepsilon}_2 = \phi^{-1}\theta_2 r,$$
$$\hat{\psi}_{21} = \psi_{21} - \phi^{-1}\theta_2\theta_1 = -\phi^{-1}\theta_2\theta_1 = \tfrac{2}{3},$$
$$\hat{\psi}_{22} = \psi_{22} - \phi^{-1}\theta_2^{\,2} = \tfrac{2}{3}.$$

Similarly, Node 2 also shares its local results $(r_2, \theta_2)$ with Node 1, such that the latter can also compute its reconciled estimate and covariances, as follows:

$$\hat{x}_1 = y_1 - \hat{\varepsilon}_1 = 9\tfrac{1}{3} + \tfrac{2}{3}\sqrt{2}; \text{ where } \hat{\varepsilon}_1 = \phi^{-1}\theta_1 r,$$
$$\hat{\psi}_{11} = \psi_{11} - \phi^{-1}\theta_1^{\,2} = \tfrac{2}{3},$$
$$\hat{\psi}_{12} = \psi_{12} - \phi^{-1}\theta_1\theta_2 = -\phi^{-1}\theta_1\theta_2 = \tfrac{2}{3}.$$

The exact sequence of the above processing steps can be found in Table 2.1.

Note that the resulting estimates are equivalent with those obtained in the centralized DR. This can be shown by gathering the computations and results from the two nodes and writing them in matrix form as follows. Firstly, the aggregates can be written in terms of centralized variables as follows:

$$r = r_1 + r_2 = a_{11}y_1 + a_{12}y_2 = Ay\,,$$

$$\phi = a_{11}\theta_1 + a_{12}\theta_2 = a_{11}(a_{11}\psi_{11} + a_{12}\psi_{12}) + a_{12}(a_{11}\psi_{21} + a_{12}\psi_{22}) = A\Psi A^T\,.$$

Then, the error estimates can be written in matrix form as follows:

$$\begin{bmatrix}\hat{\varepsilon}_1\\ \hat{\varepsilon}_2\end{bmatrix} = \phi^{-1}\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} r = \phi^{-1}\begin{bmatrix}(a_{11}\psi_{11} + a_{12}\psi_{12})\\ (a_{11}\psi_{21} + a_{12}\psi_{22})\end{bmatrix} r = \left(A\Psi A^T\right)^{-1}\Psi A^T Ay,$$

which is identical to the expression for error estimates of the centralized DR in (2.6), as $A\Psi A^T$ is a scalar quantity. Similarly, the estimate covariances can also be written in matrix form as:

$$\begin{bmatrix}\hat{\psi}_{11} & \hat{\psi}_{12}\\ \hat{\psi}_{21} & \hat{\psi}_{22}\end{bmatrix} = \begin{bmatrix}\psi_{11} & \psi_{12}\\ \psi_{21} & \psi_{22}\end{bmatrix} - \begin{bmatrix}\theta_1^2 & \theta_1\theta_2\\ \theta_2\theta_1 & \theta_2^2\end{bmatrix}\phi^{-1} = \Psi - \left(\Psi A^T\right)\left(A\Psi\right)\left(A\Psi A^T\right)^{-1},$$

which, similarly, is identical to the expression for the estimate covariances of the centralized DR in (2.8).

Furthermore, note that the local results $\theta_p$ and $r_p$ ($p$ = 1 for Node 1, and 2 for Node 2) are actually the basic building blocks in the computation of the DR solutions

above. Through the coordination algorithms in the distributed DR, the nodes share, aggregate and use the building blocks to obtain the final DR solutions.

Table 2.1: Example 1: Distributed DR processing in a basic two-node network

| Step | Snapshots of the network estimates | Node 1, *p = 1* | Node 2, *p = 2* |
|---|---|---|---|
| 0 | $\hat{x} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$<br><br>$\hat{\Psi} = \begin{bmatrix} \sigma_1^{\,2} & \\ & \sigma_2^{\,2} \end{bmatrix}$ | Own measurement and measurement variance:<br>$y_1 = 10 + \sqrt{2}, \sigma_1 = 1$<br><br>Relationship with Node 2:<br>Constraint:<br>$a_{11}x_1 + a_{12}x_2 = x_1 - x_2 = 0$<br>$(a_{11} = 1, a_{12} = -1)$<br><br>Covariances:<br>$\psi_{11} = \sigma_1^{\,2}, \psi_{12} = 0$ | Own measurement and measurement variance:<br>$y_2 = 8, \sigma_2 = \sqrt{2}$<br><br>Relationship with Node 1:<br>Constraint:<br>$a_{11}x_1 + a_{12}x_2 = x_1 - x_2 = 0$<br>$(a_{11} = 1, a_{12} = -1)$<br><br>Covariances:<br>$\psi_{22} = \sigma_2^{\,2}, \psi_{21} = 0$ |
| 1 | | Node 1 starts the reconciliation process:<br>Computes local results:<br>$r_1 = a_{11}y_1 = 10 + \sqrt{2}$<br>$\theta_1 = (a_{11}\psi_{11} + a_{12}\psi_{12}) = 1$<br><br>Sends local results $(r_1, \theta_1)$ to Node 2, such that Node 2 can compute its estimates. | |
| 2 | | | Receives local results $(r_1, \theta_1)$ from Node 1<br><br>Computes own local results:<br>$r_2 = a_{12}y_2 = -8$<br>$\theta_2 = (a_{11}\psi_{21} + a_{12}\psi_{22}) = -2$<br><br>Computes the aggregates using combined local results:<br>constraint residual:<br>$r = a_{11}y_1 + a_{12}y_2 = r_1 + r_2 = 2 + \sqrt{2}$<br><br>residual variance:<br>$\phi = a_{11}(a_{11}\psi_{11} + a_{12}\psi_{12}) + a_{12}(a_{11}\psi_{21} + a$<br>$= a_{11}\theta_1 + a_{12}\theta_2 = 1 + 2 = 3$<br><br>Sends completed aggregates $(r, \phi)$ and local result $\theta_2$ to Node 1 so that Node 1 can calculate its own reconciled estimates |
| 3 | $\hat{x} = \begin{bmatrix} 9\frac{1}{3} + \frac{2}{3}\sqrt{2} \\ 9\frac{1}{3} + \frac{2}{3}\sqrt{2} \end{bmatrix}^T$<br>$\hat{\Psi} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} \end{bmatrix}$ | Receives completed aggregates $(r, \phi)$ and local result $\theta_2$ from Node 2.<br><br>Computes reconciled estimates using the aggregates and local results: | Computes reconciled estimates using the aggregates and local results: |

| | | | |
|---|---|---|---|
| | | $\hat{\varepsilon}_1 = \phi^{-1}(a_{11}\psi_{11} + a_{12}\psi_{12})r$<br>$= \phi^{-1}\theta_1 r$<br>$= \frac{2}{3} + \frac{1}{3}\sqrt{2}$<br>$\hat{x}_1 = y_1 - \hat{\varepsilon}_1$<br>$= 9\frac{1}{3} + \frac{2}{3}\sqrt{2}$<br>$\hat{\psi}_{11} = \psi_{11} - \phi^{-1}(a_{11}\psi_{11} + a_{12}\psi_{12})^2$<br>$= \psi_{11} - \phi^{-1}\theta_1^{\,2} = \frac{2}{3}$<br>$\hat{\psi}_{12} = \psi_{12} - (\phi^{-1}(a_{11}\psi_{11} + a_{12}\psi_{12})\ldots$<br>$\times (a_{11}\psi_{21} + a_{12}\psi_{22}))$<br>$= \psi_{12} - \phi^{-1}\theta_1\theta_2$<br>$= \frac{2}{3}$ | $\hat{\varepsilon}_2 = \phi^{-1}(a_{11}\psi_{21} + a_{12}\psi_{22})r$<br>$= \phi^{-1}\theta_2 r$<br>$= -\frac{4}{3} - \frac{2}{3}\sqrt{2}$<br>$\hat{x}_2 = y_2 - \hat{\varepsilon}_2$<br>$= 9\frac{1}{3} + \frac{2}{3}\sqrt{2}$<br>$\hat{\psi}_{21} = \psi_{21} - (\phi^{-1}(a_{11}\psi_{21} + a_{12}\psi_{22})\ldots$<br>$\times (a_{11}\psi_{11} + a_{12}\psi_{12}))$<br>$= \psi_{21} - \phi^{-1}\theta_2\theta_1$<br>$= \frac{2}{3}$<br>$\hat{\psi}_{22} = \psi_{22} - \phi^{-1}(a_{11}\psi_{21} + a_{12}\psi_{22})^2$<br>$= \psi_{22} - \phi^{-1}\theta_2^{\,2}$<br>$= \frac{2}{3}$ |

## Reconstruction of failed/missing nodes:

The distributed DR is robust to node failure, while the centralized scheme is vulnerable to the failure of its central processing node. This can be illustrated by the two-node network in Example 1. When Node 2 fails, Node 1 can give sub-optimal estimates for itself and, through the constraint matrix *A,* an estimate for Node 2. In this case, it uses its own measurement $y_1$ as the best sub-optimal estimate $\hat{x}_1$, and uses the constraint relation *A* to obtain the best sub-optimal estimate of Node 2, $\hat{x}_2$:

$$\hat{x}_1 - \hat{x}_2 = 0 \Rightarrow \hat{x}_2 = \hat{x}_1 = y_1 = 10 + \sqrt{2}\,.$$

In contrast, in the centralized scheme, when Node 2 fails, Node 1 cannot do anything to alleviate the situation as it does not have the necessary knowledge of its relationship with Node 2 and is not programmed to do any processing other than sending its sensor readings. DR for the whole network is therefore disabled in this situation.

It should be noted that Node 2 is one of several possible central processing locations common in practice. In the process industry, the practice is for all sensors to send their data to a dedicated application controller/SCADA. In sensor networks, a base station located in the network can be equipped with more energy and computational and communication resources to gather data from all sensors and process them. Similarly, in the above example, Node 2 can be assumed to be a more powerful sensor node tasked with the central DR processing. The computation steps and requirements in the above example are the same regardless of the location of the central processor.

While Example 1 gives a basic idea of the distributed DR, the DR problem

considered here is not general enough to provide a complete illustration of the proposed framework. The DR problem in Example 1 contains only one constraint. In general, a DR problem contains more measured variables and more than one constraints, which means more complex computation as compared to that of a single constraint problem. As a result, additional steps need to be introduced in the distributed DR algorithm. To show this, Example 2 is constructed in the following sub-section.

### 2.3.3 Example 2: A three-node network (see Figure 2.2)

Nodes 1 and 2 are related through the constraint $x_1 - x_2 = 0$, while Nodes 2 and 3, through the constraint $x_2 - x_3 = 0$. The corresponding constraint matrix *A* is shown in Figure 2.2. Node 3 is within the communication range of Node 1 and vice versa, but no explicit constraint relationship is defined between them. Given the measurements of the three sensors:

$$y = [y_1 \quad y_2 \quad y_3]^T = [10 + \sqrt{2} \quad 8 \quad 13 - 3\sqrt{2}]^T,$$

with their corresponding variances:

$$\sigma_1^{\,2} = 1,\ \sigma_2^{\,2} = 2,\ \sigma_3^{\,2} = 3,$$

the DR problem can again be stated as (2.5), where in this case,

$$\varepsilon = [\varepsilon_1 \quad \varepsilon_2 \quad \varepsilon_3]^T = [y_1 - x_1 \quad y_2 - x_2 \quad y_3 - x_3]^T,$$

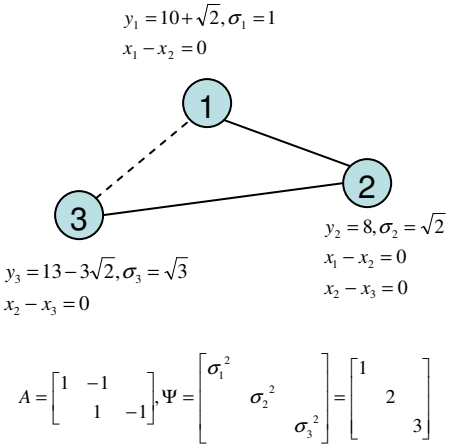



Figure 2.2. A three-node example

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} 1 & -1 & \\ & 1 & -1 \end{bmatrix},$$

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} = \begin{bmatrix} \sigma_1^{\,2} & & \\ & \sigma_2^{\,2} & \\ & & \sigma_3^{\,2} \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 2 & \\ & & 3 \end{bmatrix}.$$

Note that the sub-network consisting of Node 1, Node 2 and the constraint relating the two nodes (i.e. $x_1 - x_2 = 0$) is identical to the two-node network in Example 1. Hence, as will be seen later, the computational steps involved in data reconciliation between Node 1 and Node 2 will be identical to those of Example 1.

## Centralized DR:

Similar to Example 1, assume that the processing node is Node 2. The DR solutions are calculated by Node 2 using (2.6)-(2.8), giving:

$$\hat{\varepsilon} = \begin{bmatrix} \sqrt{2} & -2 & 3-3\sqrt{2} \end{bmatrix}^T,$$

$$\hat{x} = \begin{bmatrix} 10 & 10 & 10 \end{bmatrix}^T,$$

and

$$\hat{\Psi} = \begin{bmatrix} \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \end{bmatrix}.$$

Similar to Example 1, in order to do the above calculations, Node 2 must have the values of $y_1$, $y_2$, $y_3$, $\sigma_1$, $\sigma_2$, $\sigma_3$ and $A$, which means that Nodes 1 and 3 have to send a number of their sensor readings to Node 2, and that the constraint $A$ must be stored in Node 2.

## Distributed DR:

Table 2.2 details the steps involved for the distributed DR processing. Each node is initialized with its relevant information as seen in Step 0 of Table II, which is similar to Step 0 of Table 2.1.

The node with the lowest index, Node 1, initiates the distributed DR. Node 1 starts by reconciling with Node 2 using the constraint $x_1 - x_2 = 0$, i.e. the first row of matrix $A$ (steps 1-3 in Table 2.2). As the steps involved in the processing of this row of $A$ are identical to those shown in Table 2.1 of Example 1, and in the interest of brevity, only the final results are shown in Table 2.2.

After reconciling with Node 2, Node 1 has finished processing its constraint. Node

2 then continues the distributed DR by processing the next constraint, $x_2 - x_3 = 0$. To process this constraint, Node 2 needs to liaise with Node 3, as the latter is also involved in the constraint. Since Node 2 now has the reconciled estimate $\hat{x}_2^{(1)}$ and estimate covariances $\hat{\psi}_{21}^{(1)}, \hat{\psi}_{22}^{(1)}$, where the superscript $^{(1)}$ denotes that these estimates are obtained from processing the first row of *A*, it will use these data to reconcile with Node 3. Steps 4-6 comprise procedures for data reconciliation between Nodes 2 and 3. They are similar to steps 1-3 of Tables 2.1 and 2.2, so only the final results are shown in Table 2.2.

After steps 4-6 are completed, additional steps are necessary to update the reconciled estimates and estimate covariances of Node 1. This is because Node 1 has been correlated with Node 2 after the first row of A is processed, i.e. $\hat{\psi}_{21}^{(1)} = \frac{2}{3} \neq 0$, such that Node 1 needs to be updated whenever Node 2 is updated. The steps needed for Node 1 to compute its estimates are shown in steps 7-9 of Table 2.2. Firstly, Node 2 will send relevant data from processing row 2 of *A*, including $(r, \phi, \theta_2, \theta_3, a_{22}, a_{23})$, to Node 1 (Step 7 of Table 2.2). Then, using the data received from Node 2, Node 1 proceeds to compute its estimates (Step 8 of Table 2.2):

$$\theta_1 = a_{22}\hat{\psi}_{1,2}^{(1)} + a_{23}\hat{\psi}_{13}^{(1)} = a_{22}\hat{\psi}_{12}^{(1)} = \tfrac{2}{3} \ \left(\text{since } \hat{\psi}_{13}^{(1)} = 0 \text{ at Node 1}\right),$$
$$\hat{x}_1 = \hat{x}_1^{(1)} - \hat{\varepsilon}_1 = 10; \text{ where } \hat{\varepsilon}_1 = \phi^{-1}\theta_1 r,$$
$$\hat{\psi}_{11} = \hat{\psi}_{11}^{(1)} - \phi^{-1}\theta_1^{\,2} = \tfrac{6}{11},$$
$$\hat{\psi}_{12} = \hat{\psi}_{12}^{(1)} - \Delta\hat{\psi}_{12} = \tfrac{6}{11}; \text{ where } \Delta\hat{\psi}_{12} = \phi^{-1}\theta_1\theta_2,$$
$$\hat{\psi}_{13} = \hat{\psi}_{13}^{(1)} - \Delta\hat{\psi}_{13} = -\Delta\hat{\psi}_{13} = \tfrac{6}{11}; \text{ where } \Delta\hat{\psi}_{13} = .\phi^{-1}\theta_1\theta_3.$$

After Node 1 finishes computing its estimates, it will send the covariance updates $\hat{\psi}_{12}$ and $\hat{\psi}_{13}$ to Nodes 2 and 3, respectively (Steps 8 and 9 of Table 2.2).

After Nodes 2 and 3 finish processing all its constraints (Step 9 of Table 2.2) the distributed DR processing is therefore complete for this particular set of measurements $y$.

Note that the final estimates are also identical with those obtained in the centralized scheme. As the constraint matrix $A$ in this example has more than one rows, the algebraic proof of the equivalence between the distributed DR and the centralized scheme involves proving that processing the rows of $A$ in the sequential manner shown above will give the same results as processing the whole matrix $A$ simultaneously as in (2.6) and (2.8). This is shown in Appendix 2B.

Table 2.2 : Example 2: distributed DR processing in a three-node network

| Step | Snapshots of the network estimates | Node 1 | Node 2 | Node 3 |
|---|---|---|---|---|
| 0 | $\hat{x} = [y_1 \quad y_2 \quad y_3]^T$<br><br>$\hat{\Psi} = \begin{bmatrix} \sigma_1^2 & & \\ & \sigma_2^2 & \\ & & \sigma_3^2 \end{bmatrix}$ | Own measurement and measurement variance:<br>$y_1 = 10 + \sqrt{2}, \sigma_1 = 1$<br><br>Relationship with neighbours:<br>Neighbours:<br>$N_1 = \{x_2, \quad x_3\}$<br><br>Constraints:<br>$x_1 - x_2 = 0$<br><br>Covariances:<br>$\psi_{11} = \sigma_1^2, \psi_{12} = 0, \psi_{13} = 0$ | Own measurement and measurement variance:<br>$y_2 = 8, \sigma_2 = \sqrt{2}$<br><br>Relationship with neighbours:<br>Neighbours:<br>$N_2 = \{x_1, \quad x_3\}$<br><br>Constraints:<br>$x_1 - x_2 = 0$<br>$x_2 - x_3 = 0$<br><br>Covariances:<br>$\psi_{21} = 0, \psi_{22} = \sigma_2^2, \psi_{23} = 0$ | Own measurement and measurement variance:<br>$y_3 = 13 - 3\sqrt{2}, \sigma_3 = \sqrt{3}$<br><br>Relationship with neighbours:<br>Neighbours :<br>$N_3 = \{x_1, \quad x_2\}$<br><br>Constraints:<br>$x_2 - x_3 = 0$<br><br>Covariances:<br>$\psi_{31} = 0, \psi_{32} = 0, \psi_{33} = \sigma_3^2$ |
| 1-3 | $\hat{x} = \begin{bmatrix} 9\frac{1}{3} + \frac{2}{3}\sqrt{2} \\ 9\frac{1}{3} + \frac{2}{3}\sqrt{2} \\ y_3 \end{bmatrix}$<br><br>$\hat{\Psi} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} & \\ \frac{2}{3} & \frac{2}{3} & \\ & & \sigma_3^2 \end{bmatrix}$ | After processing constraint $x_1 - x_2 = 0$,<br>Reconciled estimate:<br>$\hat{x}_1 = 9\frac{1}{3} + \frac{2}{3}\sqrt{2}$<br><br>Estimate covariances:<br>$\hat{\psi}_{11} = \frac{2}{3}, \hat{\psi}_{12} = \frac{2}{3}$<br><br>Node 1 finishes processing its constraint. | After processing constraint $x_1 - x_2 = 0$,<br>$\hat{x}_2 = 9\frac{1}{3} + \frac{2}{3}\sqrt{2}$<br><br>Estimate covariances:<br>$\hat{\psi}_{21} = \frac{2}{3}, \hat{\psi}_{22} = \frac{2}{3}$ | |

| | | | | |
|---|---|---|---|---|
| 4-6 | $\hat{x} = \begin{bmatrix} 9\frac{1}{3} + \frac{2}{3}\sqrt{2} \\ 10 \\ 10 \end{bmatrix}$<br><br>$\hat{\Psi} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} & \\ \frac{2}{3} & \frac{6}{11} & \frac{6}{11} \\ & \frac{6}{11} & \frac{6}{11} \end{bmatrix}$ | | Node 2 continues the distributed DR. The constraint $x_1 - x_2 = 0$ has previously been processed. So, Node 2 starts processing its next constraint, $x_2 - x_3 = 0$.<br><br>After Node 2 & Node 3 reconciles using constraint $x_2 - x_3 = 0$,<br>Reconciled estimate:<br>$\hat{x}_2 = 10$<br><br>Estimate covariances:<br>$\hat{\psi}_{22} = \frac{6}{11}, \hat{\psi}_{23} = \frac{6}{11}$ | After Node 2 & Node 3 reconciles using constraint $x_2 - x_3 = 0$,<br>Reconciled estimate:<br>$\hat{x}_3 = 10$<br><br>Estimate covariances:<br>$\hat{\psi}_{33} = \frac{6}{11}, \hat{\psi}_{32} = \frac{6}{11}$ |
| 7 | | | However, Node 2 is correlated to Node 1 through previous constraint, $x_1 - x_2 = 0$, i.e.:<br>$\hat{\psi}_{21} = \frac{2}{3} \neq 0$<br><br>Therefore, Node 2 sends the following to Node 1 so that the latter can update its covariances: $\boxed{\begin{array}{l} x_2 - x_3 = 0 \\ r = -\frac{11}{3} + \frac{11}{3}\sqrt{2}, \phi = \frac{11}{3} \\ \theta_2 = \frac{2}{3}, \theta_3 = -3 \end{array}}$ | |
| 8 | | Receives data from Node 2.<br><br>Uses the received data to update its reconciled estimate and covariances with Node 2 and 3:<br>Own reconciled estimate & estimate | | |

| | | | | |
|---|---|---|---|---|
| | | variance:<br>$\theta_1 = (\psi_{12})(1) + (\psi_{13})(-1) = \frac{2}{3}$<br>$\hat{\varepsilon}_1 = \phi^{-1}\theta_1 r = -\frac{2}{3} + \frac{2}{3}\sqrt{2}$<br>$\hat{x}_1 = \hat{x}_1 - \hat{\varepsilon}_1 = 10$<br>$\hat{\psi}_{11} = \hat{\psi}_{11} - \phi^{-1}{\theta_1}^2 = \frac{6}{11}$<br><br>$\hat{\psi}_{12} = \hat{\psi}_{12} - \Delta\hat{\psi}_{12} = \frac{6}{11}$;<br>where $\Delta\hat{\psi}_{12} = \phi^{-1}\theta_1\theta_2 = \frac{4}{33}$<br><br>$\hat{\psi}_{13} = \hat{\psi}_{13} - \Delta\hat{\psi}_{13} = \frac{6}{11}$;<br>where $\Delta\hat{\psi}_{13} = \phi^{-1}\theta_1\theta_3 = -\frac{6}{11}$<br><br>Then, sends the covariance updates $\Delta\hat{\psi}_{1,2}$ and $\Delta\hat{\psi}_{1,3}$ to Node 2 and Node 3, respectively. | | |
| 9 | $\hat{x} = [10 \quad 10 \quad 10]^T$<br><br>$\hat{\Psi} = \begin{bmatrix} \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \end{bmatrix}$ | | Receives covariance update from $\Delta\hat{\psi}_{1,2}$ from Node 1:<br>Updates covariance accordingly:<br>$\hat{\psi}_{12} = \hat{\psi}_{12} - \Delta\hat{\psi}_{12} = \frac{2}{3} - \frac{4}{33} = \frac{6}{11}$<br><br>Now, constraint $x_2 - x_3 = 0$ has finished processing; Node 2 has finished processing all its constraints. | Receives covariance update from $\Delta\hat{\psi}_{1,3}$ from Node 1:<br>Updates covariance accordingly:<br>$\hat{\psi}_{1,3} = \hat{\psi}_{1,3} - \Delta\hat{\psi}_{1,3} = 0 - \left(-\frac{6}{11}\right) = \frac{6}{11}$ |

## Reconstruction of failed/missing nodes

Now consider the case when Node 2 fails. Then similar to Example 1, the centralized DR will be totally disabled, while the distributed DR will be able to provide best sub-optimal estimates using the remaining nodes, Nodes 1 and 3. The reconstruction of the estimates of Node 2 by Nodes 1 and 3 proceeds as follows, with details shown in Table 2.3. Node 1 first starts to reconcile with Node 2 through the constraint $x_1 - x_2 = 0$ (first row of A), but discovers that Node 2 has failed. Similar to Example I, Node 1 then assigns its own measurement and measurement variance as its best sub-optimal estimate and estimate variance, respectively, and uses the constraint $x_1 - x_2 = 0$ to reconstruct the estimate of Node 2 as follows:

$$\hat{x}_1 - \hat{x}_2 = 0 \Rightarrow \hat{x}_2 = \hat{x}_1 = 10 + \sqrt{2}\,. \tag{2.9}$$

The estimate variance of Node 2 and estimate covariances between Nodes 1 and 2 can be calculated based on the relationship between $\hat{x}_1$ and $\hat{x}_2$ in (2.9), i.e.

$$\begin{aligned} &\hat{x}_2 = \hat{x}_1 \\ &\Rightarrow \mathrm{cov}(\hat{x}_2) = \mathrm{cov}(\hat{x}_1) = 1 \\ &\Rightarrow \mathrm{cov}(\hat{x}_1, \hat{x}_2) = \mathrm{cov}(\hat{x}_1) = 1 \end{aligned} \quad .$$

Since Node 1 has no more constraint to process, it notifies Node 3 that Node 2 has failed, and sends the current reconstructed estimates of Node 2 to Node 3.

Node 3 continues the distributed DR by looking at its constraint $x_2 - x_3 = 0$. This constraint involves Node 2, but since reconstructed estimates of Node 2 are available,

they can be used to process the constraint. The steps involved in processing the constraint is similar to steps 4-6 of Table 2.2 as well, just that steps that are supposed to be performed by Node 2 is performed by Node 3 itself using the reconstructed estimates of Node 2. The results are shown in steps 2-4 of Table 2.3.

Additional steps similar to steps 7-9 of Table 2.2 are also necessary here, because Node 2 has been correlated with Node 1 after it is initially reconstructed by Node 1. This is reflected in the covariance between Nodes 1 and 2, i.e. $\hat{\psi}_{12} = 1 \neq 0$. Therefore, Node 3 needs to send relevant data to Node 1 such that it can update its estimates accordingly. Again, in the interest of brevity, only the final results are shown in steps 5-7 of Table 2.3.

Node 3 is the node with highest index in the network, so the distributed DR is completed, and the best sub-optimal estimates for all nodes have been obtained. Note that both neighbours of Node 2, i.e. Node 1 and Node 3, keep the reconstructed estimates of Node 2.

Table 2.3 : Example 2: reconstruction of a missing node in the three-node network

| Step | Snapshots of the network estimates | Node 1 | Node 2 | Node 3 |
|---|---|---|---|---|
| 0 | $\hat{x} = [y_1 \quad ? \quad y_3]^T$<br><br>$\hat{\Psi} = \begin{bmatrix} \sigma_1^2 & & \\ & ? & \\ & & \sigma_3^2 \end{bmatrix}$ | Own measurement and measurement variance:<br>$y_1 = 10 + \sqrt{2}, \sigma_1 = 1$<br><br>Relationship with neighbours:<br>Neighbours:<br>$N_1 = \{x_2, \quad x_3\}$<br><br>Constraints:<br>$x_1 - x_2 = 0$<br><br>Covariances:<br>$\psi_{11} = \sigma_1^2, \psi_{12} = 0, \psi_{13} = 0$ | Own measurement and measurement variance:<br>$y_2 = ?, \sigma_2 = ?$ | Own measurement and measurement variance:<br>$y_3 = 13 - 3\sqrt{2}, \sigma_3 = \sqrt{3}$<br><br>Relationship with neighbours:<br>Neighbours :<br>$N_3 = \{x_1, \quad x_2\}$<br><br>Constraints:<br>$x_2 - x_3 = 0$<br><br>Covariances:<br>$\psi_{31} = 0, \psi_{32} = 0, \psi_{33} = \sigma_3^2$ |
| 1 | $\hat{x} = \begin{bmatrix} 10 + \sqrt{2} \\ 10 + \sqrt{2} \\ y_3 \end{bmatrix}^T$<br><br>$\hat{\Psi} = \begin{bmatrix} 1 & 1 & \\ 1 & 1 & \\ & & \sigma_3^2 \end{bmatrix}$ | Node 1 starts to reconcile with Node 2 through the constraint $x_1 - x_2 = 0$ . However, Node 1 finds out that Node 2 has failed.<br><br>Node 1 then assign its own measurement and variance as its best sub-optimal estimate and estimate variance, respectively.<br>$\hat{x}_1 = y_1 = 10 + \sqrt{2}$<br>$\hat{\psi}_{11} = \sigma_1^2 = 1$<br><br>Using $x_1 - x_2 = 0$ , Node 1 is able to reconstruct the estimate of Node 2: | | |

| | | | | |
|---|---|---|---|---|
| | | Reconstructed estimate of Node 2 and the relevant estimate covariances:<br>$\hat{x}_2 = \hat{x}_1 = 10 + \sqrt{2}$<br>$\hat{\psi}_{22} = \sigma_1^{\,2} = 1$<br>$\hat{\psi}_{12} = \sigma_1^{\,2} = 1$<br><br>Node 1 has finished processing all the rows of *A* that it stores. Node 1 notifies Node 3 that Node 2 has failed, and passes the reconstructed estimates of Node 2 to Node 3.<br>$\hat{x}_2 = 10 + \sqrt{2}$<br>$\hat{\psi}_{22} = 1$<br>$\hat{\psi}_{12} = 1$ | | |
| 2-4 | $\hat{x} = \begin{bmatrix} 10+\sqrt{2} \\ 10\frac{3}{4} \\ 10\frac{3}{4} \end{bmatrix}^T$<br><br>$\hat{\Psi} = \begin{bmatrix} 1 & 1 & \\ 1 & \frac{3}{4} & \frac{3}{4} \\ & \frac{3}{4} & \frac{3}{4} \end{bmatrix}$ | | | Node 3 receives notification that Node 2 has failed and the reconstructed estimates of Node 2 from Node 1.<br><br>Node 3 starts reconciling using the constraint $x_2 - x_3 = 0$ and the reconstructed estimates of Node 2, in the similar way as it would if Node 2 were available. The results of the reconciliation are:<br>Reconciled estimate and estimate covariances of Node 3:<br>$\hat{x}_3 = 10\frac{3}{4}$<br>$\hat{\psi}_{32} = \frac{3}{4}, \hat{\psi}_{33} = \frac{3}{4}$<br><br>Updated reconstructed estimate of Node 2 and its covariances:<br>$\hat{x}_2 = 10\frac{3}{4}$<br>$\hat{\psi}_{22} = \frac{3}{4}, \hat{\psi}_{23} = \frac{3}{4}$ |

| | | | | |
|---|---|---|---|---|
| 5-7 | | Node 1 receives data from processing the constraint $x_2 - x_3 = 0$ and updates its estimate and covariances:<br>$\hat{x}_1 = 10\frac{3}{4}$<br>$\hat{\psi}_{11} = \frac{3}{4}, \hat{\psi}_{12} = \frac{3}{4}, \hat{\psi}_{13} = \frac{3}{4}$<br><br>It also sends the covariance updates $\Delta\hat{\psi}_{1,2}$ and $\Delta\hat{\psi}_{1,3}$ to Node 3.<br><br>In addition, it also keeps the updated reconstructed estimate and covariances of Node 2 received from Node 1. | | As Node 2 is correlated with Node 1, i.e. $\hat{\psi}_{12} = 1 \neq 0$, Node 3 sends data from processing the current constraint $x_2 - x_3 = 0$ to Node 1 and asks Node 1 to update its covariances.<br><br>In addition, because Node 1 is also a neighbour of Node 2, Node 3 also sends the updated reconstructed estimate and covariances of Node 2 to Node 1.<br><br>After receiving the covariance updates from Node 1, the final updated covariances are:<br>$\hat{\psi}_{2,1} = \frac{3}{4}$, $\hat{\psi}_{3,1} = \frac{3}{4}$ |

### 2.3.4 The general *N*-node network

At this point, the reader should already be acquainted with the distributed DR as applied to simple two-node and three-node networks. The general distributed DR for an *N*-node network is implemented by the coordination protocol in Appendix 2A. The coordination protocol is presented in the form of algorithm for a sensor node with index *p* (where $p = 1 \ldots N$ ) in an *N*-node network.

For a particular network, for example, the 3-node network in Example 2, by implementing the algorithm in Appendix 2A on each Nodes 1, 2 and 3 with $p = 1$, $p = 2$ and $p = 3$ , respectively, and initializing each node with appropriate data/information, Table 2.2 can be reproduced. Furthermore, if Node 2 fails in the 3-node network, the algorithm in Appendix 2A will give the same steps as Table 2.3. Of course, some programming details have been omitted in Tables 2.1–2.3 to improve clarity.

The mathematical basis of the distributed DR algorithm is the sequential DR shown in Appendix 2B. Therefore, in the distributed DR the constraint matrix *A* is processed row by row in a sequential manner. The general distributed DR algorithm for an *N*-node network can be summarized as follows. For each row of the constraint matrix *A*, the distributed processing in the sensor network is done in the following steps. First, each node *p* that is directly involved, i.e. has non-zero coefficients, in the current row of *A* performs local computations involving its own data. Second, the aggregate quantities $\phi$ and $r$ are computed from the local computation results collected from all the directly involved nodes through some communication scheme between the nodes. Third, the aggregate quantities are sent to all directly involved nodes through some

propagation mechanism. Fourth, using the aggregate quantities, each directly involved node can then compute its own reconciled estimate and its covariance with the other directly involved nodes. Fifth, a collaboration mechanism is executed by involved nodes that have correlations with some nodes that are not directly involved in the current row, to compute the updated reconciled estimate of these indirectly involved nodes and the estimate covariance between these nodes with the other directly involved nodes. The reconciled estimates after completing the five steps then become the input for processing the next row of *A*, and the cycle continues until all rows of *A* have been processed.

The algorithm for the general *N*-node network is mathematically equivalent to the DR solution in Equations (2.6)-(2.8), as has been demonstrated in the two-node example (Example 1, page 14-15) and discussed in the three-node example (Example 2, page 22-23), as well as mathematically proven in Appendix 2B. Therefore, after all nodes have taken part in the DR processing, the resulting solutions will be identical to those of the centralized DR algorithm. This applies for any number *N* of nodes in the network.

A key advantage of DDR/DBE is its ability to degrade gracefully when nodes fail. The effect of node failures on the performance of the DDR/DBE can be described as follows. When nodes fail, they do not contribute to DR/BE anymore, hence causing the final reconciled estimates to have larger variances. To see this, one can revisit equation 2.17 (Appendix 2B) on the estimate covariances after each row of the constraint matrix *A* has been processed. For ease of presentation, equation 2.17 is reproduced here as follows,

$$\Psi_j = \Psi_{j-1} - \Psi_{j-1} a_{j-1}{}^T \left( a_{j-1} \Psi_{j-1} a_{j-1}{}^T \right)^{-1} a_{j-1} \Psi_{j-1}$$

As the covariance matrix $\Psi_{j-1}$ is always positive definite, the second term at the right hand side of the equality sign, is always positive definite as well. Therefore, the variances, which are the diagonal elements of the covariance matrix, are always reduced after each row of the matrix *A* is processed.

Now, if any node fails, the row(s) of *A* in which it is involved can no longer be used in the reconciliation (they are instead used to reconstruct the estimate of the failed node). Consequently, the reduction of variances contributed by these rows of *A* are also lost, resulting in the final reconciled estimates having higher variances compared to the estimates if the node has not failed.

Therefore, the performance of DDR/DBE can be measured by the estimate variances. Furthermore, an upper threshold for the variance can be defined to assess whether the performance is acceptable. As a numerical example, the 3-node network in Example 2 for two scenarios, i.e. where all nodes are working and where Node 2 fails, will be compared as follows.

The measurement variances of the three sensor nodes are given as

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & & \\ & \sigma_2^2 & \\ & & \sigma_3^2 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 2 & \\ & & 3 \end{bmatrix}.$$

When all three nodes are working, the DR proceeds as normal for the three nodes. After Nodes 1 and 2 reconcile, the overall estimate covariance becomes:

$$\hat{\Psi} = \begin{bmatrix} \frac{2}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

It is seen from the main diagonal of $\hat{\Psi}$ that the variances of the estimates for Nodes 1 and 2 are smaller than their corresponding measurement variances.

Next, after Nodes 2 and 3 reconcile, and after the covariances of Node 1 have been updated accordingly, the overall estimate covariance becomes:

$$\hat{\Psi} = \begin{bmatrix} \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \\ \frac{6}{11} & \frac{6}{11} & \frac{6}{11} \end{bmatrix}.$$

It is seen from the main diagonal of $\hat{\Psi}$ that the variances of estimates for Nodes 1 and 2 have become smaller than before Nodes 2 and 3 reconcile. Also, again, the variance of the estimate for Node 3 is smaller than its corresponding measurement variance.

In the scenario where Node 2 fails, however, Node 1 first reconstructs the estimate of Node 2 $\hat{x}_2$ through the constraint $x_1 - x_2 = 0$ using its own measurement as the best estimate for $\hat{x}_1$, such that the estimate for Node 2 follows that of (the measurement of ) Node 1. This means that after the reconstruction, the overall estimate covariance matrix $\hat{\Psi}$ becomes

$$\hat{\Psi} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}.$$

Note that as there is no reconciliation, the variance of the estimate for Node 1 does not improve compared to its measurement variance.

Next, Node 3 receives the reconstructed estimate of Node 2, and proceeds to reconcile with Node 2 using the reconstructed estimate. The resulting estimate covariance matrix, after Node 1 has been updated accordingly, is:

$$\hat{\Psi} = \begin{bmatrix} \frac{3}{4} & \frac{3}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{3}{4} & \frac{3}{4} \\ \frac{3}{4} & \frac{3}{4} & \frac{3}{4} \end{bmatrix},$$

which shows that the variances of the estimates for all three nodes have improved compared to the measurement variances. However, these variances are still larger than the variances when all three nodes are working.

Although diagonal covariance matrix is used in the example, the DR and therefore, DDR, do not assume diagonality of the covariance matrix. If the measurement noises are correlated, one could specify a general, non-diagonal covariance matrix $\Psi$. Using the specified non-diagonal $\Psi$, the DDR proceeds with the same steps as when $\Psi$ is diagonal (see example in Appendix 2C).

From Table 2.1, the amount of information exchange for the distributed DR can be computed as two transmission instances for the 2-node network. For the 3-node network, the amount of information exchange can also be computed from Table 2.2 as seven transmission instances. In general, the total number of transmission instance is a function of connectivity of the network as reflected in matrix *A*. For conventional centralized DR, however, the number of transmission instances is equal to twice the number of nodes in the network, i.e. the sum of transmissions by all nodes to and

from the central processing node. It should also be noted that the actual amount of information exchange in the real deployment will be a function of many factors such as the network structure, network size/ transmission distance, and so on. Exact quantification would therefore be a challenge for future work.

## 2.4 Conclusion

In this chapter, Distributed Data Reconciliation (DDR) is derived, and an implementation algorithm is given. The proposed DDR enables a group of intelligent sensors to perform DR in-network. By applying DDR, each sensor node is made aware of itself and its neighbours, and is intelligent enough to respond to abnormal situation such as a missing/failed neighbouring node. The dependence on a central processing node to perform DR is eliminated, making the proposed DDR robust to the failure of the central processing node. The proposed DDR can hence be seen as a step towards realizing the paradigm of autonomous intelligent sensor networks. The proposed distributed DR in its current form lays the groundwork for further exploration in distributing more elaborate processing of sensor measurements. A good candidate is the treatment of biases in the measurements, which is the subject of Chapter 4.

# Appendix 2A

## Algorithm for a sensor node in the general *N*-node network

Algorithm for a sensor node in the general *N*-node network

**On event: Initialize():**

$p \leftarrow node\_index$ //index of the node

$J_p \leftarrow \{j; a_{j,p} \neq 0\}$ //indices of constraints in which it is involved

FOR EACH $j \in J_p$

$E_{p,j} \leftarrow \{q; a_{j,q} \neq 0\}$ //neighbours related through constraints

$A_{p,j} \leftarrow \{(q, a_{j,q}); q \in E_{p,j}\}$ //constraint coefficients

$\hat{\psi}_{pq}; q \in E_{p,j} \leftarrow 0$ //covariances of estimates

$y_p \leftarrow 0$ //initialize sensor measurement

$\sigma_p \leftarrow initial\_sensor\_variance$

$missing\_list \leftarrow \{\}$//list of missing nodes

$\hat{J}, J_s \leftarrow \{\}$ //processed constraints, skipped constraints due to missing nodes

$\hat{\psi}_{p,p} \leftarrow \sigma_p$

$\tilde{r}, \tilde{\phi} \leftarrow 0$

$\Theta, E \leftarrow \{\}$

NewData();

IF $p = lowest\_index$,

RcvToken();

**On event: Aggregate():**

$r_p \leftarrow a_{j,p} y_{j,p}$

$\theta_p \leftarrow \sum_{k \in E_{p,j}} \hat{\psi}_{j,pk} a_{j,k}$

$\tilde{r} \leftarrow \tilde{r} + r_p$

$\tilde{\phi} \leftarrow \tilde{\phi} + a_{j,p} \theta_p$

$\Theta = \Theta \cup \{(p, \theta_p)\}$

$E = E \cup \{(p,q); \hat{\psi}_{pq} \neq 0, a_{jq} = 0, (p,q) \notin E\}$

IF $p = \max(E_{p,j})$ //highest index in the constraint

$r \leftarrow \tilde{r}$

$\phi \leftarrow \tilde{\phi}$

Propagate();

ELSE

agg_pax $\leftarrow \{j, \tilde{r}, \tilde{\phi}, \Theta, E, missing\_list\}$

$\tilde{E}_{p,j} \leftarrow \{q; q > p, q \in E_{p,j}\}$

$q \leftarrow \min(\tilde{E}_{p,j})$

SendCmd('Aggregate', agg_pax) to Node $q$;

**On event: Propagate():**

$\tilde{E}_{p,j} \leftarrow \{q; q < p, q \in E_{p,j}\}$

IF $\tilde{E}_{p,j} \neq \{\ \}$

$q \leftarrow \max(\tilde{E}_{p,j})$

prop_pax $\leftarrow \{j, r, \phi, \Theta, E, missing\_list\}$

SendCmd('Propagate', prop_pax) to Node $q$;

Reconcile();

IF any $p$ in $(p,q) \in E$ , // i.e. it has indirectly involved neighbour,

FOR EACH $(p,q) \in E$ ,

coll_pax $\leftarrow \{j, r, \phi, \Theta, E, missing\_list, A_{p,j}\}$

SendCmd('Collaborate', coll_pax) to $q$

$\hat{J} \leftarrow \hat{J} \cup \{j\}$ //append current constraint to the list of processed constraints

IF $\tilde{E}_{p,j} = \{\ \}$ //last node in the constraint to finish reconciliation = node with the DR token

RcvToken();

**On event: Collaborate():**

$E_j \leftarrow \{k; a_{j,k} \neq 0\}$

$\theta_p \leftarrow \sum_{k \in E_j} \hat{\psi}_{j,pk} a_{j,k}$

$\hat{\varepsilon}_p \leftarrow \phi^{-1} \theta_p r_p$

$\hat{x}_p \leftarrow \hat{x}_p - \hat{\varepsilon}_p$

FOR EACH $q \in E_j$ //directly involved nodes

$\Delta\hat{\psi}_{pq} \leftarrow \phi^{-1} \theta_p \theta_q$

$update\_type \leftarrow$'cov'

covUpdate $\leftarrow \{update\_type, \Delta\hat{\psi}_{pq}\}$

SendCmd('UpdateCov', covUpdate) to Node $q$

$\hat{\psi}_{pq} \leftarrow \hat{\psi}_{pq} - \Delta\hat{\psi}_{pq}$

$E_i \leftarrow \{q; (s,q) \in E, s \in E_j\}$

FOR EACH $q \in E_{p,j}$ //indirectly involved nodes

$update\_type \leftarrow$'$ie$'

covUpdate $\leftarrow \{update\_type, \theta_p, j\}$

SendCmd('UpdateCov', covUpdate) to Node $q$

**On event: UpdateCov():**

IF *update_type* = 'cov'

$\hat{\psi}_{pq} \leftarrow \hat{\psi}_{pq} - \Delta\hat{\psi}_{pq}$

ELSEIF *update_type* = 'ie'

$\Delta\hat{\psi}_{pq} \leftarrow \phi^{-1}\theta_p\theta_q$

$\hat{\psi}_{pq} \leftarrow \hat{\psi}_{pq} - \Delta\hat{\psi}_{pq}$

$update\_type \leftarrow$'cov'

covUpdate $\leftarrow \{update\_type, \Delta\hat{\psi}_{pq}\}$

SendCmd('UpdateCov', covUpdate) to Node $q$

**On event: Reconstruct():**

IF *recon_type* = 'data-request'

$recon\_type \leftarrow$ 'data-update'

recon_pax $\leftarrow \{\hat{x}_p, \sigma_p, recon\_type\}$

SendCmd('Reconstruct', recon_pax) to Node $q$

ELSEIF *recon_type* = 'data-update'

keep the received data $\{\hat{x}_q, \sigma_q\}$ for use in reconstruction

ELSEIF *recon_type* = 'estimate-update'

IF is a neighbour of a solved missing node in $\hat{X}$

save the reconstructed value of the missing node from $\hat{X}$

update processed constraint list & skipped constraint list

Pass the recon_pax to the next highest index relevant neighbours

**On event: RcvToken():**

IF $missing\_list \neq \{\}$

//update the constraint list for reconstruction:

$R = \{(j, A_{p,j}); a_{j,p} \neq 0, a_{j,q} \neq 0, \forall j \in J_p, \forall q \in missing\_list\}$

$X \leftarrow \{\}$

IF $R \neq \{\}$

$X \leftarrow \{(p, \hat{x}_p)\}$

$\tilde{R} \leftarrow \tilde{R} \cup R - \left(\tilde{R} \cap R\right)$

$\tilde{X} \leftarrow \tilde{X} \cup X$

//try to reconstruct

IF info in $\tilde{R}$ is sufficient to solve for any node in *missing_list*,

FOR each relevant node $q$, //request data from other relevant nodes

$recon\_type \leftarrow$ 'data-request'

recon_pax $\leftarrow \{recon\_type\}$

SendCmd('Reconstruct', recon_pax) to Node $q$

recon_pax $\leftarrow \{\}$

WaitForData()

solve for the missing node

add the solution to $\hat{X}$

remove the solved missing/biased node from *missing_list* , accordingly

update processed constraint list and skipped constraint list

FOR each relevant node $q$, // update relevant nodes on reconstructed estimates & //processed constraint list

$recon\_type \leftarrow$ 'estimate-update'

recon_pax $\leftarrow \{\tilde{R}, \tilde{X}, \hat{X}, missing\_list, recon\_type\}$

SendCmd('Reconstruct', recon_pax) to Node $q$

ELSE

update skipped constraint list

IF is a neighbour of a solved missing node in $\hat{X}$
save the reconstructed value of the missing node from $\hat{X}$

$\tilde{J} \leftarrow J_p - \hat{J} - J_s$
IF $\tilde{J} \neq \{\}$, //if still have unprocessed constraint
$j \leftarrow \min\{\tilde{J}\}$
Aggregate();
ELSE,
$\tilde{E}_p \leftarrow \bigcup_{j \in J_p} \{q; q \in E_{p,j}, q > p\}$
IF $\tilde{E}_p \neq \{\}$
$q \leftarrow \min\{\tilde{E}_p\}$
SendCmd('RcvToken') to Node $q$
ELSE
ReverseToken(); //one round of DR is done, so return token to lowest index

**On event: ReverseToken():**
$\tilde{E}_p \leftarrow \bigcup_{j \in J_p} \{q; q \in E_{p,j}, q < p\}$
IF $\tilde{E}_p \neq \{\}$
$q \leftarrow \max\{\tilde{E}_p\}$
SendCmd('ReverseToken') to Node $q$
NewData();
ELSE
NewData();
RcvToken();

**On event: ReceiveCmd(command, dataPacket):**
IF command = 'Acknowledged'
Acknowledged();
ELSE
SendCmd('Acknowledged') to <Node p>
update local info with info in <dataPacket>
do <command> using <dataPacket>

**Acknowledged():**
$ack \leftarrow 1$;

**SendCmd(command, dataPacket):**
$ack \leftarrow 0$; //assume only one TimeOut is active at any point of time
send <command> and <dataPacket> to <Node q>
IF command $\neq$ 'Acknowledged'
WaitForAck();

**WaitForAck():**
WHILE NOT time_out,
wait //wait for ack from node to which it sent cmd
IF $ack = 0$,
$missing\_list \leftarrow missing\_list \cup \{q\}$

SkipConstraint(*j*);

**WaitForData():**

WHILE NOT time_out,
wait //wait for data from node to which it sent request
IF recon_pax = { },
$missing\_list \leftarrow missing\_list \cup \{q\}$
SkipConstraint(*j*);

**SkipConstraint(*j*):**

$J_s \leftarrow J_s \cup \{j\}$ //add *j* to list of skipped constraints
SendCmd('RcvToken') to lowest index in $E_{p,j}$

**NewData():**

$y_p \leftarrow new\_sensor\_reading$
$\sigma_p \leftarrow updated\_sensor\_variance$

**Reconcile():**

$\hat{\varepsilon}_p \leftarrow \Phi^{-1}\theta_p r_p$
$\hat{x}_p \leftarrow \hat{x}_p - \hat{\varepsilon}_p$
FOR EACH $q \in E_{p,j}$
$\Delta\hat{\psi}_{pq} \leftarrow \phi^{-1}\theta_p\theta_q$
$\hat{\psi}_{pq} \leftarrow \hat{\psi}_{pq} - \Delta\hat{\psi}_{pq}$

# Appendix 2B

## Proof of equivalence between *batch* and *sequential* DR

*Batch* DR is when the constraint matrix $A$ is processed simultaneously, in contrast to *sequential* DR where the constraint matrix $A$ is processed row by row (or several rows at a time). The prrof of equivalence between batch and sequential DR is as follows. For the DR problem in (2.5), partition the constraint matrix into $A = \begin{bmatrix} a_1 \\ A_2 \end{bmatrix}$, where $a_1$ is the first row of $A$, and $A_2$ contains the rest of the rows of $A$. Now, formulate a new DR problem using $a_1$ as the constraint matrix. From (2.6), the solutions to this new DR problem can be expressed as

$$\text{error estimate: } \hat{\varepsilon}_1 = \Psi a_1^T (a_1 \Psi a_1^T)^{-1} a_1 y, \tag{2.10}$$

$$\text{estimate covariance: } \hat{\Psi}_1 = \Psi - \Psi a_1^T (a_1 \Psi a_1^T)^{-1} a_1 \Psi. \tag{2.11}$$

Next, formulate another new DR problem using $A_2$ as the constraint matrix. However, instead of the measurement vector $y$ and measurement covariance matrix $\Psi$, the reconciled estimate $\hat{x}_1 = y - \hat{\varepsilon}_1$ and estimate covariance matrix $\hat{\Psi}_1$ from the first DR problem are used in the second DR problem. From (2.6), the solution of this DR problem can be expressed as:

$$\text{error estimate: } \hat{\varepsilon}_2 = \Psi_2 A_2^T (A_2 \Psi_2 A_2^T)^{-1} A_2 (y - \hat{\varepsilon}_1); \text{ where } \Psi_2 = \hat{\Psi}_1, \tag{2.12}$$

$$\text{estimate covariance: } \hat{\Psi}_2 = \Psi - \Psi a_1^T (a_1 \Psi a_1^T)^{-1} a_1 \Psi. \tag{2.13}$$

The sequence of processing the two DR problems above is called *sequential* DR. In contrast, the *batch* DR is the original DR problem using the unpartitioned constraint matrix *A*. Therefore, the equivalence between the sequential and batch DR holds if the final solution of the sequential DR is equal to the solution of the batch DR, i.e.

$$\hat{\varepsilon} = \hat{\varepsilon}_1 + \hat{\varepsilon}_2, \tag{2.14}$$

$$\hat{\Psi} = \hat{\Psi}_2, \tag{2.15}$$

where $\hat{\varepsilon}$ is as expressed in (2.6), while $\hat{\varepsilon}_1$ and $\hat{\varepsilon}_2$ are as expressed in (2.10) and (2.12) respectively.

To prove the equivalence, let the batch solution be computed as follows.

$$\begin{aligned}\hat{\varepsilon} &= \Psi A^T (A\Psi A^T)^{-1} Ay \\ &= \Psi \begin{bmatrix} a_1^T & A_2^T \end{bmatrix} \left( \begin{bmatrix} a_1 \\ A_2 \end{bmatrix} \Psi \begin{bmatrix} a_1^T & A_2^T \end{bmatrix} \right)^{-1} \begin{bmatrix} a_1 \\ A_2 \end{bmatrix} y \\ &= \begin{bmatrix} \Psi a_1^T & \Psi A_2^T \end{bmatrix} \begin{bmatrix} a_1 \Psi a_1^T & a_1 \Psi A_2^T \\ A_2 \Psi a_1^T & A_2 \Psi A_2^T \end{bmatrix}^{-1} \begin{bmatrix} a_1 y \\ A_2 y \end{bmatrix}\end{aligned} \tag{2.16}$$

Denote $\begin{bmatrix} a_1 \Psi a_1^T & a_1 \Psi A_2^T \\ A_2 \Psi a_1^T & A_2 \Psi A_2^T \end{bmatrix}^{-1}$ as $\begin{bmatrix} P & Q \\ R & S \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$.

Solving $\begin{bmatrix} P & Q \\ R & S \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$ for E, F, G and H yields:

$$E = P^{-1} + P^{-1}Q\left(S - RP^{-1}Q\right)^{-1} RP^{-1}$$
$$F = -P^{-1}Q\left(S - RP^{-1}Q\right)^{-1}$$
$$G = -\left(S - RP^{-1}Q\right)^{-1} RP^{-1}$$
$$H = \left(S - RP^{-1}Q\right)^{-1}$$

Substituting the values of P, Q, R and S according to (2.16) and simplifying yields (2.14), hence proving the equivalence of the error estimates. Similar procedure can be performed for the estimate covariance of the batch DR, resulting in (2.15).

By recursively breaking up $A_2$ into two, the above proof can be extended to any number $m$ of rows of $A$, resulting in:

$$\hat{\varepsilon} = \sum_{j=1}^{m} \hat{\varepsilon}_j;$$
$$\hat{x}_j = \hat{x}_{j-1} - \hat{\varepsilon}_j = y - \sum_{k=1}^{j} \hat{\varepsilon}_k;$$
where:
$$\hat{\varepsilon}_j = \Psi_j a_j{}^T \left(a_j \Psi_j a_j{}^T\right)^{-1} a_j \hat{x}_{j-1};$$
$$\Psi_j = \Psi_{j-1} - \Psi_{j-1} a_{j-1}{}^T \left(a_{j-1} \Psi_{j-1} a_{j-1}{}^T\right)^{-1} a_{j-1} \Psi_{j-1};$$
$$\hat{x}_0 = y;$$
$$\Psi_0 = \Psi;$$

hence proving that the sequential DR results in the same solution as the batch DR.

It is seen that with sequential DR, the original (batch) DR problem becomes a series of single-constraint DR problems. An interesting implication is that for the sequential DR, the condition for the overall constraint matrix $A$ can be slightly relaxed, i.e. $A$ can be rank deficient, but its rows must be consistent. In this case, when a row that is linearly dependent with a (some) previously processed row(s) is being processed, the resulting estimates will not change. However, $rank(A)$ must still be less than the number of sensors $N$.

# Appendix 2C

## Example: A two-node network with correlated measurement noises

Consider a sensor network consisting of two sensor nodes, related through a constraint $x_1 - x_2 = 0$ (Figure 2.1). Given the measurements of the two sensors:

$$y = [y_1 \quad y_2]^T = [10 + \sqrt{2} \quad 8]^T,$$

with their corresponding non-diagonal covariance matrix:

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} \\ \psi_{21} & \psi_{22} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \mathrm{cov}(x_1, x_2) \\ \mathrm{cov}(x_2, x_1) & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$$

The DR problem can be stated as (2.5), where in this case,

$$\varepsilon = [\varepsilon_1 \quad \varepsilon_2]^T = [y_1 - x_1 \quad y_2 - x_2]^T,$$

$$A = [a_{11} \quad a_{12}] = [1 \quad -1],$$

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} \\ \psi_{21} & \psi_{22} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \mathrm{cov}(x_1, x_2) \\ \mathrm{cov}(x_2, x_1) & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}.$$

### Centralized DR:

In a centralized scheme, all processing is done at a single location. Assume that in this example, the single location is at Node 2. The reconciled estimates can be

calculated by Node 2 in a centralized manner using (2.6)-(2.8), resulting in

$$\hat{\varepsilon} = \begin{bmatrix} \frac{4}{5} + \frac{2}{5}\sqrt{2} & -1\frac{1}{5} - \frac{3}{5}\sqrt{2} \end{bmatrix}^T,$$

$$\hat{x} = \begin{bmatrix} 9\frac{1}{5} + \frac{3}{5}\sqrt{2} & 9\frac{1}{5} + \frac{3}{5}\sqrt{2} \end{bmatrix}^T,$$

and

$$\hat{\Psi} = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} \end{bmatrix}.$$

In order to carry out the above calculations, Node 2 must have the values of $y_1$, $y_2$, $\sigma_1$, $\sigma_2$, $\mathrm{cov}(x_1, x_2)$ and $A$. The measurement $y_1$ is usually an average of readings taken by the sensor in Node 1, while $\sigma_1$ is the variance of the readings. This means Node 1 has to send a number of its sensor readings to Node 2, so that Node 2 can compute $y_1$ and $\sigma_1$. Also, the constraint matrix $A$ must be stored in Node 2, and the covariance between the two sensors must be known to Node 2.

## Distributed DR:

In the proposed distributed DR, instead of Node 2 doing all the processing, Node 1 and Node 2 share the processing and communicate to complete the data reconciliation. Table 2.1 (which can be found at the end of this chapter) shows the details. Node 1 and 2 each computes and holds its own measurement average and variance, and keeps its own constraint and covariance relating it with each other.

To compute the reconciled estimates and estimate covariances, the nodes need data

from each other, in addition to their own local data. To do this, the nodes go through a series of computation and communication procedures as follows:

*(iii)Computation of local results using local node data:*

Node 1 computes, locally:

$$r_1 = a_{11} y_1 = 10 + \sqrt{2},$$
$$\theta_1 = a_{11}\psi_{11} + a_{12}\psi_{12} = 2.$$

Node 2 computes, locally:

$$r_2 = a_{12} y_2 = -8,$$
$$\theta_2 = a_{11}\psi_{21} + a_{12}\psi_{22} = -3.$$

*(iv)Aggregation of local results to compute reconciled estimates and covariances:*

After local computation, Node 1 sends its local results $(r_1, \theta_1)$ to Node 2, which then aggregates them with its local results as follows:

$$r = r_1 + r_2 = 2 + \sqrt{2},$$
$$\phi = a_{11}\theta_1 + a_{12}\theta_2 = 5.$$

With these aggregate values, Node 2 can now compute its own reconciled estimate and covariances, as follows:

$$\hat{x}_2 = y_2 - \hat{\varepsilon}_2 = 9\tfrac{1}{5} + \tfrac{3}{5}\sqrt{2};\ \text{where}\ \hat{\varepsilon}_2 = \phi^{-1}\theta_2 r,$$
$$\hat{\psi}_{21} = \psi_{21} - \phi^{-1}\theta_2\theta_1 = \tfrac{1}{5},$$
$$\hat{\psi}_{22} = \psi_{22} - \phi^{-1}\theta_2^{\,2} = \tfrac{1}{5}.$$

Similarly, Node 2 also shares its local results $(r_2, \theta_2)$ with Node 1, such that the latter can also compute its reconciled estimate and covariances, as follows:

$$\hat{x}_1 = y_1 - \hat{\varepsilon}_1 = 9\tfrac{1}{5} + \tfrac{3}{5}\sqrt{2};\ \text{where}\ \hat{\varepsilon}_1 = \phi^{-1}\theta_1 r,$$
$$\hat{\psi}_{11} = \psi_{11} - \phi^{-1}\theta_1^{\,2} = \tfrac{1}{5},$$
$$\hat{\psi}_{12} = \psi_{12} - \phi^{-1}\theta_1\theta_2 = \tfrac{1}{5}.$$

# Chapter 3. Application case study of DDR

## 3.1 Introduction

A case study involving a group of flow sensors in an experimental-scale chemical plant (Figure 3.1) is conducted to demonstrate the distributed data reconciliation (DDR) proposed in Chapter 2. The plant is a typical chemical reaction plant comprising two reactors with several heat exchangers and tanks.

The distributed DR is applied to five sensors measuring material flows in the plant. The scenario where one of the sensor nodes, i.e. Node 2, fails is considered. The results show that when Node 2 fails, the remaining four nodes continue the DR, and use their reconciled estimates to reconstruct the estimates of Node 2. In contrast, when Node 2 is the central processing node, the centralized DR approach fails totally when Node 2 fails.

This chapter is organized as follows. Descriptions of the plant are given in Section 3.2. The plant configuration used in the case study is described in Section 3.3. The application of DDR to the plant and the results are presented and discussed in Section

3.4. Conclusion follows in Section 3.5.

## 3.2 Plant description

The experimental-scale chemical plant studied in this chapter is built to provide a generic system plant for process control experiments. The experimental rig has been fabricated using modular design, which allows the running of numerous plant configurations. Coupled with the modular design, the control system has been designed to provide real plant control integrated with internal simulation of reaction (soft reaction) [8]. A schematic diagram of the plant is shown in Figure 3.2, showing the main operation units comprising two stirred-tank reactors, a mixer, a feed tank and a number of heat exchangers. Material feed from the feed tank is heated before being fed to the Reactor 1 and Mixer. The effluent from Reactor 1 is then mixed with the material feed in Mixer and fed to Reactor 2. The effluent from Reactor 2 is, in turn, fed back to the feed tank and the cycle continues. Each reactor uses cooling coil for temperature control.

Figure 3.1. An experimental-scale chemical reaction plant

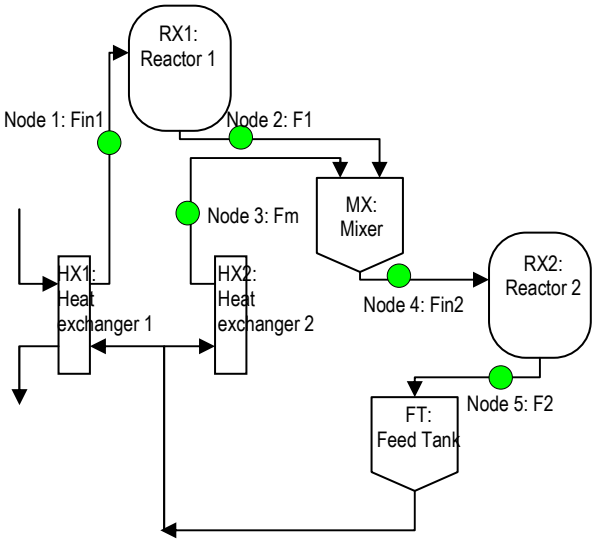
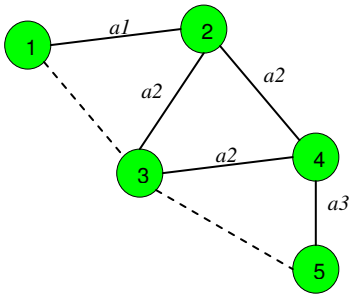



Figure 3.2. Flow diagram of the chemical reaction plant in Figure 3.1; flow sensors are in grey.

## 3.3 Experiment setup

For the case study in this chapter, five flow sensors are considered, constituting five intelligent sensor nodes. The five sensors measure feed flow to Reactor 1 ( $F_{in1}$ ), effluent flow from Reactor 1 ( $F_1$ ), feed flow to Mixer before mixing with Reactor 1 effluent ( $F_m$ ), feed flow to Reactor 2 ( $F_{in2}$ ) and effluent flow of Reactor 2 ( $F_2$ ). The location of the five flow sensors are highlighted in Figure 3.2. Steady-state relationships among the sensor nodes that constitute the DR constraints are obtained from mass balances governing the flows measured by the five sensors, i.e. $Ax = 0$, where

$$A = \begin{bmatrix} 1 & -1 & & & \\ & 1 & 1 & -1 & \\ & & & 1 & -1 \end{bmatrix},$$




$$A = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{matrix} \\ \begin{bmatrix} 1 & -1 & & & \\ & 1 & 1 & -1 & \\ & & & 1 & -1 \end{bmatrix} \end{array}$$

Figure 3.3. Diagram of sensor network for flow sensors in Figure 3.1; DR constraints resulting from mass balance equations are shown as matrix A and the lines linking the nodes.

$$x = \begin{bmatrix} F_{in1} & F_1 & F_m & F_{in2} & F_2 \end{bmatrix}^T .$$

A diagram of the sensor network, with each line representing the relationship between nodes at the two ends of the line, is shown in Figure 3.3. Similar to Figure 2.2 of Example 2 in Chapter 2, the dashed lines represent the connectivity between nodes that are not related explicitly by a row in the constraint matrix *A*. Here, Nodes 1 and 3 are within communication range of each other and so are Nodes 3 and 5.

## 3.4 Experiment & Results

Data are collected when the plant is running at steady-state. The collected flow measurements are shown in Figure 3.4 as dots and circles. Using these measurements, a scenario of node failure, i.e. Node 2, is simulated. In this scenario, from time = 1s to time = 18s, all nodes are working properly, but from time = 19s onwards, Node 2

fails. The dots in Figure 3.4 represent available measurements, while the circles in Figure 3.4(b) represent the missing measurements of Node 2.

Similar to Examples 1 and 2 in Chapter 2, each of the five nodes is initialized with the appropriate rows of the constraint matrix $A$. A time window of 15 sampling instances (15 seconds) is defined for the distributed DR. This means that the distributed DR is applied at every $15^{th}$ sampling instance. The mean and variance used in the distributed DR are calculated from the 15 measurements within a window.

For the window spanning from time = 1s to time = 15s, the distributed DR is performed at time=15s and proceeds in steps similar to Table 2.2 in Chapter 2. Node 1 will reconcile with Node 2 through row 1 of $A$, followed by Node 2 with 3 and 4 through row 2 of $A$, and finally, Node 4 with 5 through row 3 of $A$.

At the next window (time = 16s to time = 30s), however, when Node 1 starts the DR at time = 30s, it realizes that Node 2 has failed. Node 1 then uses row 1 of $A$ to reconstruct an estimate of Node 2. The reconstructed estimate is then sent to Node 3, such that the latter can use it to reconcile with Node 4 through row 2 of $A$. The reconstructed estimate of Node 2 will get updated in this reconciliation, and so Node 3 will send the updates to Node 1, who also holds reconstructed data of Node 2. Furthermore, Node 4 is also a direct neighbour of Node 2, so it will also hold the reconstructed data of Node 2. Finally, Node 4 and Node 5 reconcile through row 3 of $A$, and the distributed DR is completed for this time window.

Figure 3.4 plots the measurements and reconciled estimates of the five sensor nodes. From time = 1s to time = 15s, the reconciled estimates of all nodes are represented by solid lines. When Node 2 fails at time = 19s, its estimates for time = 16s to time 30s are still available through reconstruction by the other nodes, where they are represented by the dashed line in Figure 3.4(b). The estimates of the other,

available nodes during this time are represented by solid lines.

As a comparison, the reconciled estimates in the ideal case when all five nodes are working are also shown in Figure 3.4(a)-(e), where they are represented by the dotted lines. These are the most optimal estimates that can be given by the distributed DR. It is seen, however, that the reconstructed estimates approximate the optimal estimates very closely.

Figure 3.5 shows the reduction in estimate variances when DDR is applied to the same experimental-scale chemical reaction plant in Figure 3.1. The values on the y-axis are shown as percentages of the estimate variances with respect to the measurement variances, which are the maximum limit on the estimate variances. It is seen that the estimate variances increase when increasing number of nodes fail. The variances of the reconstructed estimates are used for failed nodes. It is seen that the estimate variances are lowest when all nodes are working. As more nodes fail, fewer nodes can reconcile among themselves, and the measurements of the nodes are used as the best estimates in cases where reconciliation cannot be done. Therefore, the variances degrade until they reach the measurement variances, where no reconciliation is possible and processing involves mainly reconstruction of failed nodes. Based on this degradation in estimate variance, one can decide and set the minimum variance reduction level at which the performance of the system is still considered acceptable.

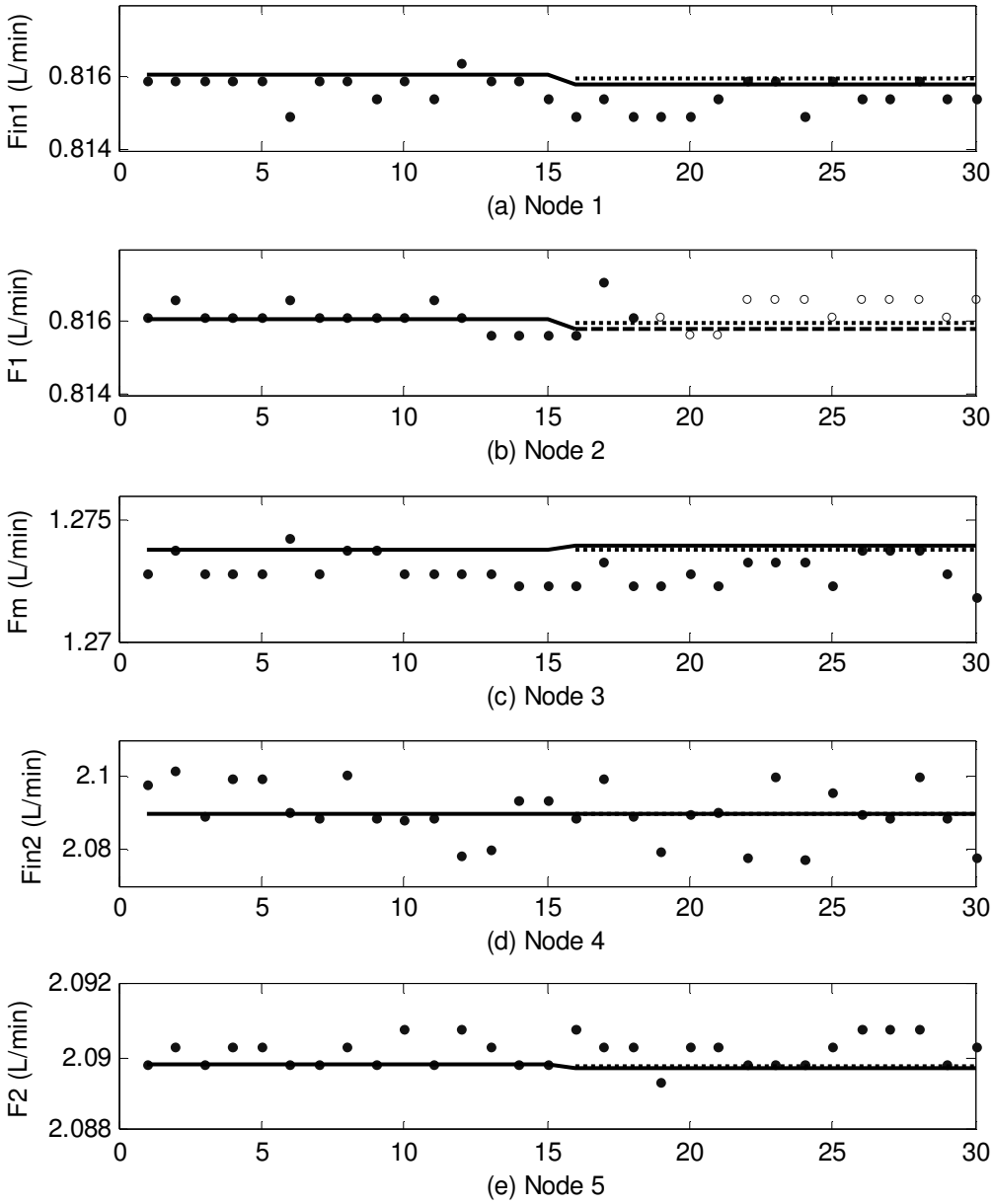



Figure 3.4. Reconstruction of a failed node: Node 2 fails at time = 19s. Dots are measurements, circles are missing measurements, solid lines are reconciled estimates, dotted lines are reconciled estimates if all nodes are working, and dashed lines are reconstructed estimates of Node 2 provided by nodes 1 and 3 through 5, showing that the reconstructed estimates are very close to the reconciled estimates had Node 2 not failed.

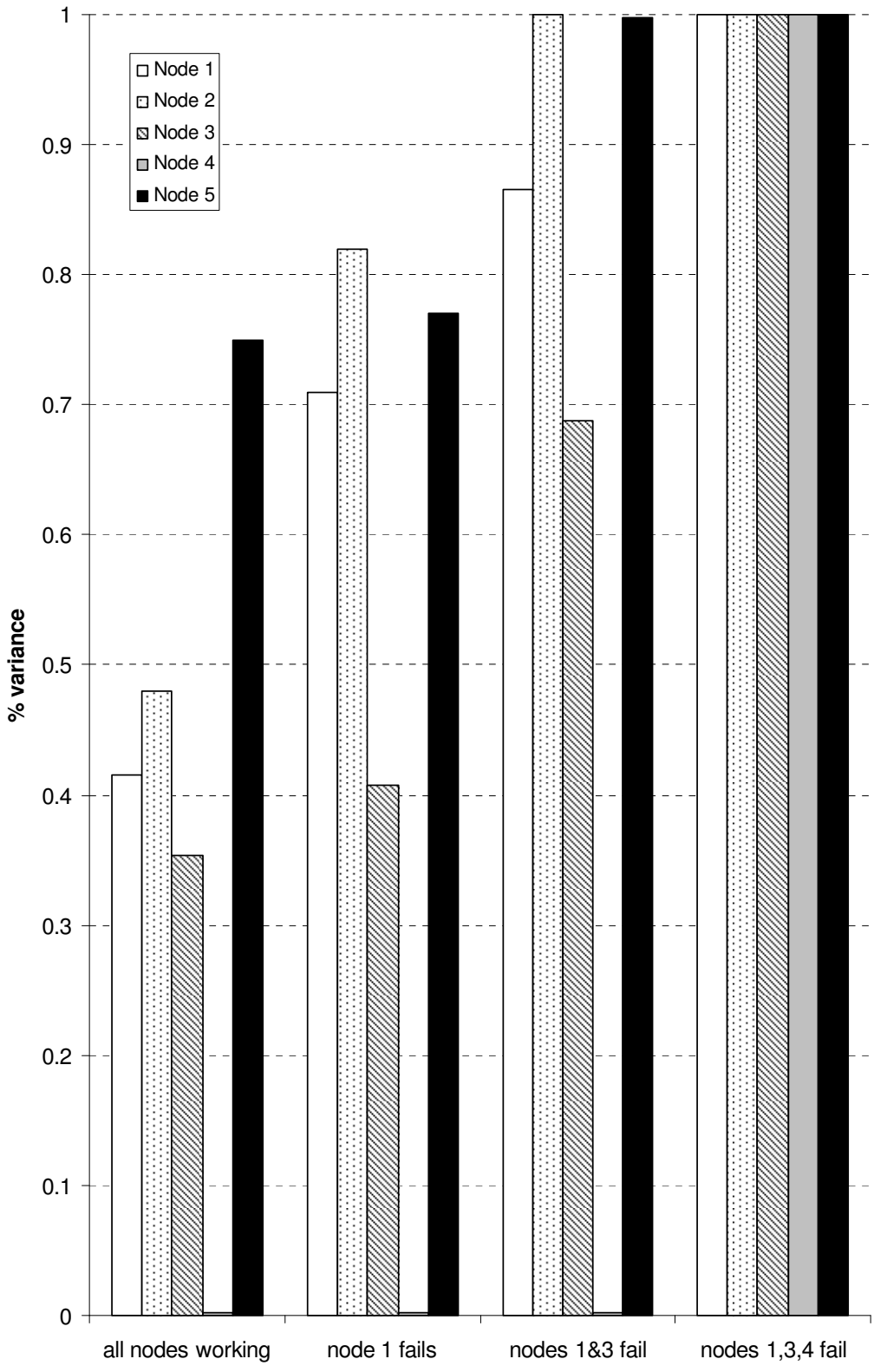



Figure 3.5. Degradation of estimate variances as increasing number of nodes fail (Note that the estimate variance of Node 4 is very small compared to its measurement variance)

## 3.5 Conclusion

The distributed data reconciliation (DDR) proposed in Chapter 2 has been applied to an experimental-scale chemical plant. The scenario where one of the sensor nodes, i.e. Node 2, fails is considered. In this scenario, from time = 1s to time = 18s, all nodes are working properly, but from time = 19s onwards, Node 2 fails. The results show that from time = 19s onwards, the remaining four nodes continue the DR, and use their reconciled estimates to reconstruct the estimates of Node 2. As a comparison, the reconciled estimates in the case where all nodes are working are also obtained. The comparison shows that the distributed DR continues to provide useful flow estimates when Node 2 fails. The case study has therefore demonstrated the capability of DDR to degrade gracefully in face of node failure.

# Chapter 4. Distributed Bias Estimation (DBE)

## 4.1 Introduction

In this chapter, instead of collecting all measurements to a central processing node to estimate their biases, the intelligence of the sensor nodes is leveraged to perform bias estimation (BE) in a distributed manner. The proposed distributed bias estimation (DBE) is robust to node failures and degrades gracefully when increasing number of nodes fail.

In considering the performance of various estimators for the bias estimation, equations relating estimator type, variance and sample size are examined. These equations enable one to compute the sample size needed by the bias estimator to meet a specified variance. Alternatively, these equations allow us to calculate the variance of the bias estimator and hence its precision if the sample size is given. Such information can be gainfully used to select appropriate bias estimators depending on applications.

GT distribution has previously been employed in econometrics [9-10] to model

random residuals in regression parameter estimation. By being a distribution superset encompassing normal, uniform, T and double exponential distributions [10], GT has the flexibility to characterize economic data with non-normal statistical properties. In process engineering, assumptions commonly made such as normal (Gaussian) statistics are approximations to reality. The occurrence of outliers, transient data in steady-state measurements, instrument failure, human error, process nature, etc. can all induce non-normal process data [11]. Indeed whenever the central limit theorem is invoked – the central limit theorem being a limit theorem can at most suggest approximate normality for real data. If the statistics of the process being monitored is not normal, conventional data reconciliation techniques may lead to poor results. GT-based data reconciliation for the state vector has been examined in a simulation study [11]. In this chapter, it is extended to bias estimation and equations are derived to relate the variance of the bias estimate to sample size.

Conventional least-squares (LS) bias estimation formulations assume that process data follows normal (Gaussian) distribution. However, even high-quality process data may not be normal. The presence of a single huge outlier can spoil the statistical analysis significantly. Hence it is not wise to use the least-squares algorithm without any built-in check [11,12]. A common practice to robustify LS is to introduce a preliminary outlier test before applying LS. In the presence of outliers, an outlier test is expected to remove some or all of them. One of the more popular tests is the interquartile (IQR) test [13].

In the light of the equations examined in this chapter, an efficient estimator can be selected. In the presence of outliers that are close to the good data, the equations show that using GT, instead of normal distribution, to characterize process data gives rise to a more efficient estimator than the LS and IQR+LS and therefore enables earlier

remedial actions against sensor bias. This will be shown in the case study in Chapter 5, in particular, Table 5.2.

The chapter is organized as follows. Section 4.2 introduces bias estimation in the centralized context, and presents analytical tools to evaluate the performance of bias estimators. Section 4.3 presents the distributed bias estimation (DBE) algorithm in detail. Lastly, Section 4.4 concludes the chapter.

## 4.2 Bias Estimation

In the following, the problem of bias estimation (BE) and its solution are stated mathematically. Let there be *N* sensors.

*Measurement model:* In the absence of bias, the *N*-dimensional measurement vector *y* is related to the *N*-dimensional state vector *x* by

$$y = x + \varepsilon ,$$

where $\varepsilon$ is the random noise vector. It is assumed that the expected value of $\varepsilon$, $E(\varepsilon) = 0$. Now, let $b$ sensors among the $N$ sensors be biased. The relationship between the measurement vector *y* and state vector *x* then becomes

$$y = x + Be + \varepsilon , \tag{4.1}$$

where e is the $b \times 1$ vector of bias magnitudes and B is the $N \times b$ vector indicating which sensors are biased. For example, if $y = [y_1 \quad y_2]^T$ represents the measurements of two sensors, where Sensor 2 is biased, $e$ will be a scalar representing the bias

magnitude of Sensor 2, while *B* will be a $2\times1$ vector:

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

*Consistency relationship model, or BE constraints:* The BE constraints consist of equations describing how the state variables $x = [x_1 \ldots x_N]^T$ are inter-related. In linear steady-state BE, it is expressed in the matrix form:

$$Ax = 0, \tag{4.2}$$

where each row of *A* is a constraint equation, i.e. a linear equation relating $x_1 \ldots x_N$.

### 4.2.1 Least squares (LS) bias estimation

The LS bias estimation algorithm is well-studied [4,17] and only the relevant equations will be given. Under the assumption that $\varepsilon$ follows multivariate normal distribution, the noise variance matrix is given by

$$\Lambda = diag(\Lambda_1, \ldots, \Lambda_N),$$

where $\Lambda_1 = \mathrm{var}(\varepsilon_i)$.

The estimation of bias vector *e* can be formulated as a minimization problem under the constraint $Ax = 0$. In the LS framework, specifically, the problem is formulated as one of finding the optimal bias estimate $e^*$ that minimizes

$$\sum_{j=1}^{S}\left(y(j)-x-Be\right)^{T}\Lambda^{-1}\left(y(j)-x-Be\right)$$

with respect to $x$ and $e$, subject to $Ax=0$, where $S$ is the sample size. Using the method of Lagrange multipliers, the solution to the above minimization problem is given by

$$e^{*}=\left(AB\right)^{-1}A\bar{y} \tag{4.3}$$

where the $i$th element of $\bar{y}$ is given by $\bar{y}_i=\frac{1}{S}\sum_{j=1}^{S}y_i(j)$ and can be rewritten as

$$\sum_{j=1}^{S}\left(y_i(j)-\bar{y}_i\right)=0. \tag{4.4}$$

If $\Omega=diag\left(\mathrm{var}(\bar{y}_1),\ldots,\mathrm{var}(\bar{y}_N)\right)$, then the variance of $e^{*}$ is calculated as

$$\Phi=(AB)^{-1}A\Omega A^{T}(B^{T}A^{T})^{-1}. \tag{4.5}$$

From (4.1), the reconciled estimate $\hat{x}$ of the state vector $x$ can then be computed as

$$\hat{x}=\bar{y}-Be^{*}. \tag{4.6}$$

It is well known that the LS algorithm is highly sensitive to outliers. A common practice to robustify LS is to introduce a preliminary outlier test before applying LS.

In the presence of outliers, and outlier test removes some or all of them. One of the more popular tests is perhaps the interquartile range (IQR) test [14,15], among others. For a given data sample, the IQR test finds the 25th percentile $Q_1$, the 75th percentile $Q_3$, and the interquartile range $IQR = Q_3 - Q_1$. Any observation $y_i(j)$ such that $y_i(j) < y_L = Q_1 - 1.5 \times IQR$ or $y_i(j) > y_H = Q_3 + 1.5 \times IQR$ will then be removed from the sample. The remaining data are then relegated to LS, i.e. (4.4) is modified to

$$\sum_{j=1}^{S'} (y_i(j) - \bar{y}_i) = 0, \quad y_L \le y_i(j) \le y_H , \tag{4.7}$$

where $S'$ is the number of the remaining data after outlier removal.

### 4.2.2 GT-based bias estimation

This estimator is based on the GT distribution, which has the probability density function

$$f_{GT}(\varepsilon; p, q, \sigma) = \frac{p}{2\sigma q^{1/p} \beta(1/p, q) \left(1 + \frac{|\varepsilon|^p}{q\sigma^p}\right)^{q+1/p}} \tag{4.8}$$

where $\varepsilon \in \Re$ is any realization value, $\sigma$ is the scaling parameter, $p$ and $q$ are the shape parameters and the beta function is given by $\beta(a,b) = \int_0^1 z^{a-1}(1-z)^{b-1} dz$. The GT distribution is unimodal and symmetric with peak at its center point, which in

(4.8) is zero. By different choices of $p$ and $q$, the GT can represent a wide range of distributions [10].

In the framework of GT, the maximum likelihood method is used to find the optimal bias estimate $e^*$ which minimizes

$$-\sum_{j=1}^{S}\sum_{i=1}^{N}\ln f_{GT}(y_i(j)-x_i-b_i e;\sigma_i,p_i,q_i) \tag{4.9}$$

with respect to $x$ and $e$, subject to $Ax=0$ and where row vector $b_i$ denotes the $i$th row of the matrix $B$.

The derivation for the solution the optimization problem in (4.9) is given in Appendix 4A. In brief, the problem can be solved by first finding $\bar{y}$ element-wise from

$$\sum_{j=1}^{S}\frac{(p_i q_i+1)\,\mathrm{sgn}(y_i(j)-\bar{y}_i)\left|y_i(j)-\bar{y}_i\right|^{p_i-1}}{q_i\sigma_i^{\,p_i}+\left|y_i(j)-\bar{y}_i\right|^{p_i}}=0 \tag{4.10}$$

and then solving for $e^*$ from

$$e^*=(AB)^{-1}A\bar{y}. \tag{4.11}$$

As with LS, if $\Omega$ is the variance of $\bar{y}$ then the variance of $e^*$ can again be calculated by (4.5). Also, similar to LS, the reconciled estimate can be computed by (4.6), i.e. $\hat{x}=\bar{y}-Be^*$.

## 4.3 Analysis of Estimator Performance

Note that from (4.5) the variances of bias estimates by LS and GT depend on the variance of $\bar{y}$ denoted by $\Omega$. Hence, to make a comparison between them, we turn to the study of $\Omega$. An important analysis tool is the Influence Function (IF) [12]. It will be used here to quantify estimate variance and to offer a qualitative picture of how outliers affect the performance of various estimators.

### 4.3.1 Influence Function (IF) and Estimator Variance

IF describes estimator behaviour in the neighbourhood of an underlying data distribution. IF quantifies the standardized effect on the resultant estimate caused by adding an extra observation to an infinite sample. It is defined as

$$IF(z) = \lim_{h\to 0} \frac{\mu((1-h)f + h\delta(z)) - \mu(f)}{h} = \frac{d}{dh}\mu((1-h)f + h\delta(z))\Big|_{h=0} \tag{4.12}$$

where $\mu(\cdot)$ is the estimator which yields an estimate given a data distribution as the input argument, $f(\varepsilon)$ is the probability density function of $\varepsilon$, and $\delta(z)$ is a unit impulse at $z$.

Consider an estimation $\mu(f)$ determined from

$$\int_{-\infty}^{+\infty} \rho(\varepsilon, \mu(f)) f(\varepsilon) d\varepsilon = 0 . \tag{4.13}$$

It follows from the assumption $E(\varepsilon) = 0$ that $\mu(f) = 0$, and

$$IF(z) = \frac{\rho(z)}{\int_{-\infty}^{+\infty}\left(\frac{d}{d\varepsilon}\rho(\varepsilon)\right)f(\varepsilon)d\varepsilon}. \tag{4.14}$$

The derivation of (4.14) can be found in Appendix 4B. Using the IF, we can approximate the variance of an estimator by

$$\text{var}(\mu(f)) \approx \frac{1}{S}\int_{-\infty}^{+\infty} IF^2(\varepsilon)f(\varepsilon)d\varepsilon, \tag{4.15}$$

where $S$ is the sample size. The derivation of (4.15) is given in Appendix 4C.

We now find the IF of LS, IQR+LS and GT based estimators using (4.14) for each Sensor $i$. To facilitate the derivation of IF we first define

$$\mu_i = \bar{y}_i - x_i - b_i e_i \tag{4.16}$$

so that together with (4.1) the term $(y_i(j) - \bar{y}_i)$ in (4.4), (4.7) and (4.10) can be substituted by

$$y_i(j) - \bar{y}_i = \varepsilon_i(j) - \mu_i. \tag{4.17}$$

### 4.3.2 IF of LS Estimator

Using (4.17), and for a continuous distribution with probability density function $f(\varepsilon)$, (4.4) becomes

$$\int_{-\infty}^{+\infty}(\varepsilon-\mu(f))f(\varepsilon)d\varepsilon=0\,, \tag{4.18}$$

where subscript *i* has been dropped for the sake of brevity. Comparing (4.13) and (4.18) gives

$$\rho(\varepsilon)=\varepsilon\,, \tag{4.19}$$

if $\mu(f)=0$. Substituting (4.19) into (4.14) gives

$$IF(z)=\frac{z}{\int_{-\infty}^{+\infty}f(\varepsilon)d\varepsilon}=z\,,$$

as $\int_{-\infty}^{+\infty}f(\varepsilon)d\varepsilon=1$.

### 4.3.3 IF of IQR+LS Estimator

Using (4.17), and for a continuous distribution with probability density function $f(\varepsilon)$, (4.7) becomes

$$\int_{-\infty}^{\varepsilon_L}0\cdot f(\varepsilon)d\varepsilon+\int_{\varepsilon_L}^{\varepsilon_H}(\varepsilon-\mu(f))f(\varepsilon)d\varepsilon+\int_{\varepsilon_H}^{+\infty}0\cdot f(\varepsilon)d\varepsilon=0\,, \tag{4.20}$$

where subscript *i* has been dropped for the sake of brevity. The limits $\varepsilon_L$ and $\varepsilon_H$ are given by

$$\varepsilon_L = Q_1 - 1.5(Q_3 - Q_1), \tag{4.21}$$

$$\varepsilon_H = Q_3 + 1.5(Q_3 - Q_1), \tag{4.22}$$

where

$$\int_{-\infty}^{Q_1} f(\varepsilon) d\varepsilon = 0.25,$$

$$\int_{-\infty}^{Q_3} f(\varepsilon) d\varepsilon = 0.75.$$

Comparing (4.13) and (4.20) gives

$$\rho(\varepsilon) = \begin{cases} \varepsilon, & \text{if } \varepsilon_L \le \varepsilon \le \varepsilon_H, \\ 0, & \text{otherwise,} \end{cases} \tag{4.23}$$

if $\mu(f) = 0$. Substituting (4.23) into (4.14) gives

$$\begin{aligned} IF(z) &= \begin{cases} z \cdot \left( \int_{-\infty}^{\varepsilon_L^+} \varepsilon_L \delta(\varepsilon_L) f(\varepsilon) d\varepsilon + \int_{\varepsilon_L^+}^{\varepsilon_H^-} f(\varepsilon) d\varepsilon - \int_{\varepsilon_H^-}^{+\infty} \varepsilon_H \delta(\varepsilon_L) f(\varepsilon) d\varepsilon \right)^{-1}, & \text{if } \varepsilon_L \le z \le \varepsilon_H, \\ 0, & \text{otherwise,} \end{cases} \\ &= \begin{cases} z \cdot \left( \varepsilon_L f(\varepsilon_L) + \int_{\varepsilon_L}^{\varepsilon_H} f(\varepsilon) d\varepsilon - \varepsilon_H f(\varepsilon_H) \right)^{-1}, & \text{if } \varepsilon_L \le z \le \varepsilon_H, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.24}$$

### 4.3.4 IF of GT based Estimator

Using (4.17), and for a continuous distribution with probability density function $f(\varepsilon)$, (4.10) becomes

$$\int_{-\infty}^{+\infty} \frac{(pq+1)\,\mathrm{sgn}(\varepsilon-\mu)\left|\varepsilon-\mu\right|^{p-1}}{q\sigma^{p}+\left|\varepsilon-\mu\right|^{p}} f(\varepsilon)d\varepsilon = 0, \tag{4.25}$$

where subscript *i* has been dropped for the sake of brevity. Comparing (4.13) and (4.25) gives

$$\rho(\varepsilon) = \frac{(pq+1)\,\mathrm{sgn}(\varepsilon)\left|\varepsilon\right|^{p-1}}{q\sigma^{p}+\left|\varepsilon\right|^{p}}, \tag{4.26}$$

if $\mu(f)=0$. Substituting (4.26) into (4.14) gives

$$IF(z_i) = \frac{(pq+1)\mathrm{sgn}(z)\left|z\right|^{p-1}}{q\sigma^{p}+\left|z\right|^{p}} \left( \int_{-\infty}^{+\infty} \frac{(pq+1)\left((p-1)\left(q\sigma^{p}+\varepsilon^{p}\right)\varepsilon^{p-2}-p\varepsilon^{2p-2}\right)}{(q\sigma^{p}+\varepsilon^{p})^{2}} f(\varepsilon)d\varepsilon \right)^{-1}. \tag{4.27}$$

### 4.3.5 Example 1: A simple 2-sensor system

In this example, the actual variance of the GT estimator is compared with the variance of the same estimator computed from (4.15), to show that (4.15) approximates the actual variance very closely. A simple 2-sensor system is considered as follows, i.e. $N=2$. Let the measured variables of the two sensors, $x_1$ and $x_2$, are related through the constraint equation $x_1 - x_2 = 0$, i.e.

$$A = \begin{bmatrix} 1 & -1 \end{bmatrix}. \tag{4.28}$$

Furthermore, let the measurement vector, state vector, bias location matrix, bias magnitude and noise matrix be, respectively,

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix},\ x = \begin{bmatrix} 10 \\ 10 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ e = 1,\ \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}, \tag{4.29}$$

where the random variables $\varepsilon_1$ and $\varepsilon_2$ have the same probability density function

$$f(\varepsilon_1) = f(\varepsilon_2) = \frac{1}{4}\delta(-0.1) + \frac{1}{2}\delta(0) + \frac{1}{4}\delta(0.1), \tag{4.30}$$

i.e. each random variable can assume the values of -0.1, 0 and +0.1 with probabilities of $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{1}{4}$, respectively.

Given these probabilities, and for a sample size of $S = 3$, the probabilities for all possible combinations of $\varepsilon_1$ and $\varepsilon_2$ and hence $y_1$ and $y_2$ are given in Table 4.1. The solution for $\bar{y}_1$ and $\bar{y}_2$ from (4.10) under $p_1 = p_2 = 2,\ q_1 = q_2 = 20,\ \sigma_1 = \sigma_2 = 0.085\sqrt{2}$ are also given. The probabilities in Table 4.1 are determined as follows. Consider $(\varepsilon_1(1), \varepsilon_1(2), \varepsilon_1(3)) = (-0.1, -0.1, 0)$. The probability is $\frac{1}{4}\times\frac{1}{4}\times\frac{1}{2} = \frac{1}{32}$. But order does not matter so $(-0.1, -0.1, 0)$, $(-0.1, 0, -0.1)$ and $(0, -0.1, -0.1)$ are considered as the same combination, and for this combination, the probability is $\left(\frac{1}{4}\times\frac{1}{4}\times\frac{1}{2}\right) + \left(\frac{1}{4}\times\frac{1}{2}\times\frac{1}{4}\right) + \left(\frac{1}{2}\times\frac{1}{4}\times\frac{1}{4}\right) = \frac{3}{32}$ as shown in Row 2.

Table 4.1: Example 1: Estimates of the means $\bar{y}_1$ and $\bar{y}_2$ for all combinations of $y_1$ and $y_2$

| $\varepsilon_1$ or $\varepsilon_2$ | | | $y_1$ | | | $y_2$ | | | Probability | $\bar{y}_1 = 10.0 + \varepsilon_1$ | $\bar{y}_2 = 11.0 + \varepsilon_2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| –0.1, | –0.1, | –0.1 | 9.9 | 9.9 | 9.9 | 10.9 | 10.9 | 10.9 | $\frac{1}{64}$ | 9.9 | 10.9 |
| – 0.1, | – 0.1, | 0 | 9.9 | 9.9 | 10.0 | 10.9 | 10.9 | 11.0 | $\frac{3}{32}$ | 9.933 | 10.933 |
| – 0.1, | – 0.1, | 0.1 | 9.9 | 9.9 | 10.1 | 10.9 | 10.9 | 11.1 | $\frac{3}{64}$ | 9.965 | 10.965 |
| – 0.1, | 0, | 0 | 9.9 | 10.0 | 10.0 | 10.9 | 11.0 | 11.0 | $\frac{3}{16}$ | 9.967 | 10.967 |
| – 0.1, | 0, | 0.1 | 9.9 | 10.0 | 10.1 | 10.9 | 11.0 | 11.1 | $\frac{3}{16}$ | 10.0 | 11.0 |
| 0, | 0, | 0 | 10.0 | 10.0 | 10.0 | 11.0 | 11.0 | 11.0 | $\frac{1}{8}$ | 10.0 | 11.0 |
| 0.1 | 0, | 0 | 10.1 | 10.0 | 10.0 | 11.1 | 11.0 | 11.0 | $\frac{3}{16}$ | 10.033 | 11.033 |
| 0.1 | 0.1 | – 0.1, | 10.1 | 10.1 | 9.9 | 11.1 | 11.1 | 10.9 | $\frac{3}{64}$ | 10.035 | 11.035 |
| 0.1 | 0.1 | 0 | 10.1 | 10.1 | 10.0 | 11.1 | 11.1 | 11.0 | $\frac{3}{32}$ | 10.067 | 11.067 |
| 0.1 | 0.1 | 0.1 | 10.1 | 10.1 | 10.1 | 11.1 | 11.1 | 11.1 | $\frac{1}{64}$ | 10.1 | 11.1 |

From Table 4.1, the expectation

$$E(\bar{y}_1) = \frac{1}{64}\times 9.9 + \frac{3}{32}\times 9.933 + \frac{3}{64}\times 9.965 + \ldots + \frac{1}{64}\times 10.1 = 10.0,$$
$$E(\bar{y}_2) = \frac{1}{64}\times 10.9 + \frac{3}{32}\times 10.933 + \frac{3}{64}\times 10.965 + \ldots + \frac{1}{64}\times 11.1 = 11.0.$$

From (4.11),

$$E(e^*) = E(\bar{y}_2) - E(\bar{y}_1) = 11.0 - 10.0 = 1.0 = e\,.$$

From Table 4.1,

$$\begin{aligned}\text{var}(\bar{y}_1) &= \frac{1}{64}\times(9.9-10.0)^2 + \frac{3}{32}\times(9.933-10.0)^2 + \ldots + \frac{1}{64}\times(10.1-10.0)^2 \\ &= 0.0017,\end{aligned}$$

$$\begin{aligned}\mathrm{var}(\bar{y}_2) &= \frac{1}{64}\times(10.9-11.0)^2+\frac{3}{32}\times(10.933-11.0)^2+\ldots+\frac{1}{64}\times(11.1-11.0)^2\\ &= 0.0017.\end{aligned}$$

The variances of $\bar{y}_1$ and $\bar{y}_2$ can also be approximated from the IF using (4.15)

$$\begin{aligned}\mathrm{var}(\bar{y}_1) = \mathrm{var}(\bar{y}_2) &\approx \frac{1}{3}\left(\frac{1}{4}IF^2(-0.1)+\frac{1}{2}IF^2(0)+\frac{1}{4}IF^2(0.1)\right)\\ &= 0.0017,\end{aligned} \tag{4.31}$$

where (4.27) gives $IF(-0.1) = -0.1016$ , $IF(0) = 0$ and $IF(0.1) = 0.1016$ . The approximate variance is close to the exact value. Equation (4.5) gives

$$\Phi = \mathrm{var}(\bar{y}_1)+\mathrm{var}(\bar{y}_2) = 0.0034\,.$$

# 4.4 Distributed Bias Estimation (DBE)

## 4.4.1 Overview

This section presents the proposed distributed bias estimation (BE) algorithm. An illustrative example of a three-node network (Section 4.4.2) is given to provide the reader a basic understanding of the algorithm and to contrast it with the centralized BE. Section 4.4.3 describes the general distributed BE for an *N*-node network, while a detailed implementation algorithm can be found in Appendix 4D.

### 4.4.2 Example 2: A three-node network

Consider a network with three sensor nodes shown in Figure 4.1. Nodes 1 and 2 are related through the constraint $x_1 - x_2 = 0$, while Nodes 2 and 3, through the constraint $x_2 - x_3 = 0$. Node 3 is within the communication range of Node 1 and vice versa, but no explicit constraint relationship between them is defined. The constraint matrix $A$ can hence be expressed as

$$A = \begin{bmatrix} 1 & -1 & \\ & 1 & -1 \end{bmatrix}. \tag{4.32}$$

Next, consider the BE problem constructed from row 1 of Table 4.2, i.e. the BE problem is expressed as the minimization in (4.9) with $S = 3$, $N = 3$, subject to the constraint $Ax = 0$, where $A$ is expressed as (4.32) and the measurements $y_1$, $y_2$ and $y_3$ are given as follows

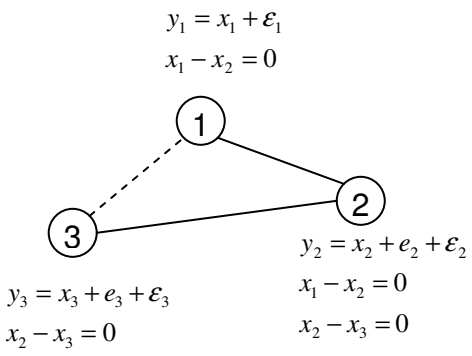



Figure 4.1. A three-node sensor network

$y_1(1) = 9.9$, $y_1(2) = 9.9$, $y_1(3) = 9.9$,

$y_2(1) = 10.9$, $y_2(2) = 10.9$, $y_2(3) = 10.9$,

$y_3(1) = 10.9$, $y_3(2) = 10.9$, $y_3(3) = 10.9$.

Note that Table 4.2 is identical to Table 4.1 except that Table 4.2 considers three sensors. Sensor 3 has the same bias as Sensor 2, so columns 4 and 8 in Table 4.2 are identical with columns 3 and 7, respectively.

Table 4.2: Example 2: Estimates of the means $\bar{y}_1$, $\bar{y}_2$ and $\bar{y}_3$ for all combinations of $y_1$, $y_2$ and $y_3$

| $\varepsilon_1$ or $\varepsilon_2$ | | | $y_1$ | | | $y_2$ | | | $y_3$ | | | Probability | $\bar{y}_1 = 10.0 + \varepsilon_1$ | $\bar{y}_2 = 11.0 + \varepsilon_2$ | $\bar{y}_3 = 11.0 + \varepsilon_3$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| −0.1, | −0.1, | −0.1 | 9.9 | 9.9 | 9.9 | 10.9 | 10.9 | 10.9 | 10.9 | 10.9 | 10.9 | $\frac{1}{64}$ | 9.9 | 10.9 | 10.9 |
| −0.1, | −0.1, | 0 | 9.9 | 9.9 | 10.0 | 10.9 | 10.9 | 11.0 | 10.9 | 10.9 | 11.0 | $\frac{3}{32}$ | 9.933 | 10.933 | 10.933 |
| −0.1, | −0.1, | 0.1 | 9.9 | 9.9 | 10.1 | 10.9 | 10.9 | 11.1 | 10.9 | 10.9 | 11.1 | $\frac{3}{64}$ | 9.965 | 10.965 | 10.965 |
| −0.1, | 0, | 0 | 9.9 | 10.0 | 10.0 | 10.9 | 11.0 | 11.0 | 10.9 | 11.0 | 11.0 | $\frac{3}{16}$ | 9.967 | 10.967 | 10.967 |
| −0.1, | 0, | 0.1 | 9.9 | 10.0 | 10.1 | 10.9 | 11.0 | 11.1 | 10.9 | 11.0 | 11.1 | $\frac{3}{16}$ | 10.0 | 11.0 | 11.0 |
| 0, | 0, | 0 | 10.0 | 10.0 | 10.0 | 11.0 | 11.0 | 11.0 | 11.0 | 11.0 | 11.0 | $\frac{1}{8}$ | 10.0 | 11.0 | 11.0 |
| 0.1 | 0, | 0 | 10.1 | 10.0 | 10.0 | 11.1 | 11.0 | 11.0 | 11.1 | 11.0 | 11.0 | $\frac{3}{16}$ | 10.033 | 11.033 | 11.033 |
| 0.1 | 0.1 | −0.1, | 10.1 | 10.1 | 9.9 | 11.1 | 11.1 | 10.9 | 11.1 | 11.1 | 10.9 | $\frac{3}{64}$ | 10.035 | 11.035 | 11.035 |
| 0.1 | 0.1 | 0 | 10.1 | 10.1 | 10.0 | 11.1 | 11.1 | 11.0 | 11.1 | 11.1 | 11.0 | $\frac{3}{32}$ | 10.067 | 11.067 | 11.067 |
| 0.1 | 0.1 | 0.1 | 10.1 | 10.1 | 10.1 | 11.1 | 11.1 | 11.1 | 11.1 | 11.1 | 11.1 | $\frac{1}{64}$ | 10.1 | 11.1 | 11.1 |

Let the measurement vector $y$, bias location matrix $B$ and measurement noise $\varepsilon$ be expressed as, respectively,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix},\ B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix},$$

where $\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$ are distributed according to (4.30), i.e.

$$f(\varepsilon_1) = f(\varepsilon_2) = f(\varepsilon_3) = \frac{1}{4}\delta(-0.1) + \frac{1}{2}\delta(0) + \frac{1}{4}\delta(0.1).$$

The GT estimator parameters for each sensor are also given by (4.21) as $p_1 = p_2 = p_3 = 2,\ q_1 = q_2 = q_3 = 20,\ \sigma_1 = \sigma_2 = \sigma_3 = 0.085\sqrt{2}$.

The bias to be estimated can be expressed as

$$e^* = \begin{bmatrix} e_2^* \\ e_3^* \end{bmatrix},$$

where $e_2^*$ and $e_3^*$ represent the biases of Nodes 2 and 3, respectively. The following will show how the three-node network computes $e^*$ using the above measurements of $y_1$, $y_2$ and $y_3$ in both centralized and distributed ways.

## Centralized BE:

In a centralized scheme, all processing is done at a single location. Assume that in

this example, the single location is at Node 2. Then Node 2 will use the given data $y_1$, $y_2$ and $y_3$ to perform all the computation steps necessary to obtain $\bar{y}_1$, $\bar{y}_2$, $\bar{y}_3$ (i.e. row 1 of Table 4.2) and to subsequently compute the biases

$$e^* = (AB)^{-1} A\bar{y} = \begin{bmatrix} \bar{y}_2 - \bar{y}_1 \\ \bar{y}_3 - \bar{y}_1 \end{bmatrix} = \begin{bmatrix} 10.9 - 9.9 \\ 10.9 - 9.9 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}. \tag{4.33}$$

Obviously, in order to do the above calculations, Node 2 as the central processing node must have the values of $y_1(k)$, $y_2(k)$ and $y_3(k)$ for $k = 1,2,3$, the GT estimator parameters $p_i, q_i, \sigma_i$ for all three sensors (i.e. $i = 1,2,3$) and all entries of the constraint matrix *A* and the bias position matrix *B*. This means that Node 1 has to send $y_1(k)$ for $k = 1,2,3$ to Node 2, so that Node 2 can compute $\bar{y}_1$. Similarly, Node 3 has to send $y_3(k)$ for $k = 1,2,3$ to Node 2, so that Node 2 can compute $\bar{y}_3$. Also, the constraint matrix *A* must be stored in Node 2.

## Distributed BE:

In the proposed distributed bias estimation, instead of Node 2 doing all the processing, Nodes 1, 2 and 3 share the processing and communicate to complete the bias estimation. Nodes 1, 2 and 3 is each initialized with its own measurements $y_i(k)$ and GT estimator parameters $p_i, q_i, \sigma_i$ ($i = 1$ for Node 1, $i = 2$ for Node 2 and $i = 3$ for Node 3) and with its own constraint that relate itself with its neighbours. In this case, Node 1 keeps row 1 of the constraint matrix *A*, Node 2 keeps both rows 1 and 2, while Node 3 keeps row 2. In addition, each node is informed about the bias status of itself and its neighbours.

The distributed processing for bias estimation is preceded by each node estimating its own mean locally, i.e. $\bar{y}_1$ by Node 1, $\bar{y}_2$ by Node 2 and $\bar{y}_3$ by Node 3, using (4.12), as follows. Node 1 solves:

$$\sum_{j=1}^{3} \frac{(p_1 q_1 + 1)\,\mathrm{sgn}\left(y_1(j) - \bar{y}_1\right)\left|y_1(j) - \bar{y}_1\right|^{p_1 - 1}}{q_1 {\sigma_1}^{p_1} + \left|y_1(j) - \bar{y}_1\right|^{p_1}} = 0$$

$$\Leftrightarrow (41)\left( \frac{\mathrm{sgn}(9.9 - \bar{y}_1)|9.9 - \bar{y}_1|}{0.289 + (9.9 - \bar{y}_1)^2} + \frac{\mathrm{sgn}(9.9 - \bar{y}_1)|9.9 - \bar{y}_1|}{0.289 + (9.9 - \bar{y}_1)^2} + \frac{\mathrm{sgn}(9.9 - \bar{y}_1)|9.9 - \bar{y}_1|}{0.289 + (9.9 - \bar{y}_1)^2} \right) = 0$$

$$\Rightarrow \bar{y}_1 = 9.9.$$

Node 2 solves:

$$\sum_{j=1}^{3} \frac{(p_2 q_2 + 1)\,\mathrm{sgn}\left(y_2(j) - \bar{y}_2\right)\left|y_2(j) - \bar{y}_2\right|^{p_2 - 1}}{q_2 {\sigma_2}^{p_2} + \left|y_2(j) - \bar{y}_2\right|^{p_2}} = 0$$

$$\Leftrightarrow (41)\left( \frac{\mathrm{sgn}(10.9 - \bar{y}_2)|10.9 - \bar{y}_2|}{0.289 + (10.9 - \bar{y}_2)^2} + \frac{\mathrm{sgn}(10.9 - \bar{y}_2)|10.9 - \bar{y}_2|}{0.289 + (10.9 - \bar{y}_2)^2} + \frac{\mathrm{sgn}(10.9 - \bar{y}_2)|10.9 - \bar{y}_2|}{0.289 + (10.9 - \bar{y}_2)^2} \right) = 0$$

$$\Rightarrow \bar{y}_2 = 10.9.$$

Similarly, Node 3 solves:

$$\sum_{j=1}^{3} \frac{(p_3 q_3 + 1)\,\mathrm{sgn}\left(y_3(j) - \bar{y}_3\right)\left|y_3(j) - \bar{y}_3\right|^{p_3 - 1}}{q_3 {\sigma_3}^{p_3} + \left|y_3(j) - \bar{y}_3\right|^{p_3}} = 0$$

$$\Leftrightarrow (41)\left( \frac{\mathrm{sgn}(10.9 - \bar{y}_3)|10.9 - \bar{y}_3|}{0.289 + (10.9 - \bar{y}_3)^2} + \frac{\mathrm{sgn}(10.9 - \bar{y}_3)|10.9 - \bar{y}_3|}{0.289 + (10.9 - \bar{y}_3)^2} + \frac{\mathrm{sgn}(10.9 - \bar{y}_3)|10.9 - \bar{y}_3|}{0.289 + (10.9 - \bar{y}_3)^2} \right) = 0$$

$$\Rightarrow \bar{y}_3 = 10.9.$$

Bias estimation then proceeds with each node taking turn to process its constraint

and reconstruct the reconciled estimate of its biased neighbour(s), if any. Firstly, Node 1, which is unbiased, assigns its own mean $\bar{y}_1$ as its reconciled estimate $\hat{x}_1$. This is in accordance to (4.6), where the first element of $\hat{x}$ can be expressed as

$$\hat{x}_1 = \bar{y}_1 - b_1 e^* = \bar{y}_1 = 9.9 ,$$

because $b_1 e^* = 0$ ($b_1$ is the first row of $B$). Node 1 then uses $\hat{x}_1$ to reconstruct the estimate of its biased neighbour, $\hat{x}_2$, through the constraint $x_1 - x_2 = 0$, i.e.

$$\hat{x}_2 = \hat{x}_1 = 9.9 . \qquad (4.34)$$

After Node 1 finishes processing its constraint, it sends the reconstructed estimate $\hat{x}_2$ to Node 2, such that the latter can estimate its bias, $e_2^*$, i.e. the first element of $e^*$. This is achieved through the use of (4.6), where the second element of $\hat{x}$ can be expressed as

$$\hat{x}_2 = \bar{y}_2 - b_2 e^* = \bar{y}_2 - e_2^* ,$$

such that $e_2^* = \bar{y}_2 - \hat{x}_2 = 1.0$ ($b_2$ is the second row of $B$ and $e_2^*$ is the first element of $e^*$).

Node 2 then continues the distributed bias estimation by processing its constraint $x_2 - x_3 = 0$. Using this constraint and its reconstructed estimate $\hat{x}_2$, Node 2 reconstructs the estimate of its biased neighbour Node 3, $\hat{x}_3$, as $\hat{x}_3 = \hat{x}_2 = 9.9$. Node 2 thus finishes processing its constraints and then sends the reconstructed estimate $\hat{x}_3$ to

Node 3.

Using its reconstructed estimate $\hat{x}_3$ received from Node 2, Node 3 can estimate its own bias as through the use of (4.6), where the third element of $\hat{x}$ can be expressed as

$$\hat{x}_3 = \bar{y}_3 - b_3 e^* = \bar{y}_3 - e_3^*,$$

such that $e_3^* = \bar{y}_3 - \hat{x}_3 = 1.0$ ($b_3$ is the second row of $B$ and $e_3^*$ is the second element of $e^*$). As the constraint $x_2 - x_3 = 0$ has already been processed, Node 3 has thus finished its processing and the distributed bias estimation is completed. Note that the resulting bias estimates $e^* = \begin{bmatrix} e_2^* & e_3^* \end{bmatrix}^T = \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}^T$ are identical with those obtained in the centralized scheme, i.e. in (4.33).

### 4.4.3 Distributed BE (DBE) for the *N*-node network

At this point, the reader should already be acquainted with the distributed BE (DBE) as applied to the simple three-node network. DBE for an *N*-node network can be implemented by the coordination protocol in Appendix 4D. The coordination protocol is written in the form of algorithm for a sensor node with index *p* (where *p* = *1…N*) in an *N*-node network.

For a particular network, for example, the 3-node network in Example 2, by implementing the algorithm in Appendix 4D on each Nodes 1, 2 and 3 with $p = 1$, $p = 2$ and $p = 3$, respectively, and initializing each with appropriate data/information, Example 3 can be reproduced. Of course, some programming details have been omitted in these examples to improve clarity.

It should be noted that although the estimator in Example 2 is based on GT, the

procedure for distributed bias estimation illustrated here also applies to LS and IQR+LS. The difference is that instead of using (4.12) to estimate $\bar{y}_i$, each Node *i* will use (4.4) to compute $\bar{y}_i$ in the case of LS; whereas in the case of IQR+LS, each Node *i* will use (4.7) to compute $\bar{y}_i$.

Lastly, note that in the event that any of the biased nodes fails, the DBE algorithm can still proceed to give the same estimates $\hat{x}$ for all three nodes. For example, in the three-node network in Example 2, if Node 2 fails, Node 1 can still compute (4.34), giving the estimate of Node 2, $\hat{x}_2$. The estimate $\hat{x}_2$ is then sent to Node 3, which can then proceed with its computation as usual. Therefore, the DBE is robust to node failure. Of course, as Node 2 has failed, its data are no longer available and hence its bias becomes irrelevant.

## 4.5 Conclusion

The distributed bias estimation is derived to enable sensor nodes to perform bias estimation in-network, and an implementation algorithm is developed. The distributed bias estimation is robust against node failure and can gracefully degrade as increasing number of nodes fail. Theoretical tools to analyze the performance of the generalized T (GT), inter-quartile range test cum least-square (IQR+LS) and least-square (LS) estimators are presented. The Influence Function (IF) is used to theoretically quantify the estimator variance. The resulting theoretical variance has been shown to be accurate through comparison with the actual variance of simulation data. As the theoretical variance is a function of the sample size used in the estimation, it can be used as a tool to select the most suitable estimator for a specified sample size, or to decide on the best sample size for a particular estimator to yield the desired

performance criterion (i.e. the variance).

# Appendix 4A

## Derivation of solution to the GT-based bias estimation problem

The bias estimation problem in the GT framework given by (4.11) is restated here:

$$\min_{x,e} - \sum_{j=1}^{S} \sum_{i=1}^{N} \ln f_{GT}(y_i(j) - x_i - b_i e; \sigma_i, p_i, q_i)$$
$$\text{s.t. } Ax = 0 \quad .$$

The solution to the GT-based bias estimation problem can be obtained by the method of Lagrange multipliers. Taking into account the constraint Ax = 0, we obtain the objective function for the Lagrange multipliers optimization problem

$$J = - \sum_{j=1}^{S} \sum_{i=1}^{N} \ln f_{GT}(y_i(j) - x_i - b_i e) + \lambda^T Ax \,,$$

where $\lambda = [\lambda_1, \ \ldots, \ \lambda_m]^T$ and $b_i$ is the *i*th row of B. GT's distributional parameters $\sigma_i, p_i, q_i$ are omitted for brevity. With the substitution $\bar{y} = x + Be$, or element-wise, $\bar{y}_i = x_i + b_i e$, the above cost function can be rewritten as

$$J = - \sum_{j=1}^{S} \sum_{i=1}^{N} \ln f_{GT}(y_i(j) - \bar{y}_i) + \lambda^T A(\bar{y} - Be) \,.$$

To minimize the objective function, equate all relevant partial derivatives to zero:

$$0 = \frac{\partial J}{\partial \bar{y}_i} = \sum_{j=1}^{S} \frac{1}{f_{GT}(y_i(j) - \bar{y}_i)} \frac{\partial f_{GT}(y_i(j) - \bar{y}_i)}{\partial \bar{y}_i} + \lambda^T a_i, \quad (i = 1, \ldots, N), \tag{4.34}$$

$$0 = \frac{\partial J}{\partial \lambda} = (\bar{y} - Be)^T A^T, \tag{4.35}$$

$$0 = \frac{\partial J}{\partial e} = -\lambda^T AB, \tag{4.36}$$

where $a_i$ is the *i*th column of A. From (4.36), $\lambda = 0$, and (4.34) reduces to

$$\sum_{j=1}^{S} \frac{1}{f_{GT}(y_i(j) - \bar{y}_i)} \frac{\partial f_{GT}(y_i(j) - \bar{y}_i)}{\partial \bar{y}_i} = 0, \ (i = 1, \ldots, N).$$

Finally, (4.35) leads to the solution for $e^*$, given by (3), i.e.

$$\hat{e} = (AB)^{-1} A\bar{y}.$$

# Appendix 4B

## Derivation of influence functions

For an estimator $\mu(f)$ whose estimation takes the form

$$\int_{-\infty}^{+\infty} \rho\left(y, \mu(f)\right) f(y) dy = 0 . \tag{4.37}$$

Replacing $f$ by $(1-h)f + hg$ gives

$$\int_{-\infty}^{+\infty} \rho\left(y, \mu\left((1-h)f + hg\right)\right)\left((1-h)f(y) + hg(y)\right) dy = 0 .$$

Differentiating both sides with respect to $h$ gives

$$\begin{aligned} 0 &= \frac{d}{dh}\left( \int_{-\infty}^{+\infty} \rho\left(y, \mu\left((1-h)f + hg\right)\right)\left((1-h)f(y) + hg(y)\right) dy \right) \\ &= \int_{-\infty}^{+\infty} \rho\left(y, \mu\left((1-h)f + hg\right)\right)\left(-hf(y) + g(y)\right) dy + \\ &\left( \int_{-\infty}^{+\infty} \frac{d\rho\left(y, \mu\left((1-h)f + hg\right)\right)}{d\mu}\left((1-h)f(y) + hg(y)\right) dy \right) \cdot \frac{d}{dh}\mu\left((1-h)f + hg\right) \end{aligned} \tag{4.38}$$

For the estimators used in this chapter, $\rho(y, \mu)$ is a function of $(y-\mu)$, i.e. $\rho(y, \mu)$ reduces to $\rho(y-\mu)$. Hence,

$$\frac{d\rho(y-\mu)}{d\mu} = \frac{d\rho(y-\mu)}{dy} .$$

Making this change in (4.38) gives

$$0 = \int_{-\infty}^{+\infty} \rho\left(y, \mu\left((1-h)f + hg\right)\right)\left(-hf(y) + g(y)\right)dy + \left(\int_{-\infty}^{+\infty} \frac{d\rho\left(y, \mu\left((1-h)f + hg\right)\right)}{dy}\left((1-h)f(y) + hg(y)\right)dy\right) \cdot \frac{d}{dh}\mu\left((1-h)f + hg\right) \quad . \tag{4.39}$$

At $h = 0$, (4.39) gives

$$\left.\frac{d}{dh}\mu\left((1-h)f + hg\right)\right|_{h=0} = \frac{\int_{-\infty}^{+\infty} \rho\left(y, \mu(f)\right)g(y)dy}{\int_{-\infty}^{+\infty}\left(\frac{d\rho\left(y, \mu(f)\right)}{dy}\right)f(y)dy}. \tag{4.40}$$

Substituting $g$ with $\delta(z)$ and using the definition of IF in (12) gives

$$\begin{aligned} IF(z) &= \left.\frac{d}{dh}\mu\left((1-h)f + h\delta(z)\right)\right|_{h=0} \\ &= \frac{\int_{-\infty}^{+\infty} \rho\left(y, \mu(f)\right)\delta(z)(y)dy}{\int_{-\infty}^{+\infty}\left(\frac{d\rho\left(y, \mu(f)\right)}{dy}\right)f(y)dy} \\ &= \frac{\rho\left(z, \mu(f)\right)}{\int_{-\infty}^{+\infty}\left(\frac{d\rho\left(y, \mu(f)\right)}{dy}\right)f(y)dy}. \end{aligned} \tag{4.41}$$

Under the assumption $\mu(f) = 0$, the IF can be simplified to

$$IF(z) = \frac{\rho(z)}{\int_{-\infty}^{+\infty}\left(\frac{d}{dy}\rho(y)\right)f(y)dy},$$

giving (14).

# Appendix 4C

## Derivation of variance of an estimator

Using Taylor series expansion

$$\mu((1-h)f+hg)|_{h=1} \approx \mu((1-h)f+hg)|_{h=0} + \frac{d}{dh}\mu((1-h)f+hg)|_{h=0} \cdot (1-0)$$

$$\mu(g) \approx \mu(f) + \frac{d}{dh}\mu((1-h)f+hg)|_{h=0} \tag{4.42}$$

By (4.41), (4.40) becomes

$$\frac{d}{dh}\mu((1-h)f+hg)\Big|_{h=0} = \int_{-\infty}^{+\infty} IF(y)g(y)dy \ . \tag{4.43}$$

By (4.43), (4.42) becomes

$$\mu(g) \approx \mu(f) + \int_{-\infty}^{+\infty} IF(y)g(y)dy \ . \tag{4.44}$$

Let a sample be $y(j)$ $(j=1,\ldots,S)$, constituting the empirical distribution

$$f_s(y) = \frac{1}{S}\sum_{j=1}^{S}\delta(y(j)),$$

where $\delta(y(j))$ is the probability density of a unit probability mass at point $y(j)$, i.e.

$\delta(y(j))$ is a unit impulse at $y(j)$. Substitute $g(y)$ with $f_S(y)$ in (4.44), and the finite sample estimator can be approximated by

$$\mu(f_S) \approx \mu(f) + \int_{-\infty}^{+\infty} IF(y) f_S(y) dy = \mu(f) + \frac{1}{S}\sum_{j=1}^{S} IF(y(j)).$$

Hence, the variance of $\mu(f_S)$ can be approximated by the variance of $\frac{1}{S}\sum_{j=1}^{S} IF(y(j))$ and $\mu_i = \bar{y}_i - x_i - e_i b_i$ as

$$\text{var}(\bar{y}) = \text{var}(\mu(f_S)) \approx \text{var}\left(\frac{1}{S}\sum_{j=1}^{S} IF(y(j))\right) = \frac{1}{S}\int_{-\infty}^{+\infty} IF^2(y) f(y) dy ,$$

which is (15).

# Appendix 4D

## Implementation protocols for a sensor node in an *N*-node network

**On event: Initialize():**

$p \leftarrow node\_index$ //index of the node

$J_p \leftarrow \{j; a_{j,p} \neq 0\}$ //indices of constraints in which it is involved

FOR EACH $j \in J_p$

$E_{p,j} \leftarrow \{q; a_{j,q} \neq 0\}$ //neighbours related through constraints

$A_{p,j} \leftarrow \{(q, a_{j,q}); q \in E_{p,j}\}$ //constraint coefficients

$\hat{\psi}_{pq}; q \in E_{p,j} \leftarrow 0$ //covariances of estimates

$bias\_list \leftarrow indices\_of\_biased\_neighbours$

$y_p \leftarrow \{\}$//initialize data set

$\Pi_p \leftarrow estimator\_parameters$ //e.g. GT estimator parameters, $\{p_{GT}, q_{GT}, \sigma_{GT}\}$

$missing\_list \leftarrow \{\}$//list of missing nodes

$\tilde{J} \leftarrow \{\}$ //accumulated constraints, for use in reconstructing biased and missing nodes

$\hat{\psi}_{p,p} \leftarrow \sigma_p$

NewWindow();

IF $p = lowest\_index$,
RcvToken();

**On event: NewWindow():**

$y_p \leftarrow new\_set\_of\_sensor\_readings$

**On event: Reconstruct():**

$combined\_list \leftarrow \bigcup\{missing\_list, bias\_list\}$

IF can solve for any in combined_list,

$\hat{X} \leftarrow solve(\tilde{J}_U)$, where $\tilde{J}_U$ = constraints relevant to reconstructed vars ;

updateCov(); //compute covariances relevant to the reconstructed variables

$\tilde{J} \leftarrow \tilde{J} - \tilde{J}_U$ //remove used constraints from accumulated constraints

SendCmd('UpdateEstimates', update_est_pax) to previous highest index node;

SendCmd('RcvToken') to next lowest index node;

ELSE
AccumulateConstraints();

**On event: UpdateEstimates();**
IF is in *biased_list* AND in *update_est_pax*
//update reconciled estimates using update_est_pax
//estimate bias using updated reconciled estimates
$1 \leftarrow$ *reconstructed*
ELSEIF is a neighbour of any node *q* in *missing_list* AND *update_est_pax*
//keep reconciled estimates of *q*
ELSE
//update relevant covariances
SendCmd('UpdateEstimates', update_est_pax) to previous highest index node;

**On event: AccumulateConstraints();**
$\tilde{J} \leftarrow \tilde{J} \cup E_p - \left(\tilde{J} \cap E_p\right)$ //add constraints to accumulated list
//update *bias_list*
SendCmd('RcvToken') to neighbour with next lowest index;

**On event: RcvToken();**
IF *highest_index*
ReverseToken();
ELSEIF not in *bias_list*
Reconstruct();
ELSE
IF *reconstructed*
Reconstruct();
ELSE
AccumulateConstraints();

**On event: ReverseToken();**
IF lowest_index
NewWindow();
RcvToken();
ELSE
SendCmd('ReverseToken') to previous highest index node;

**On event: ReceiveCmd(command, dataPacket);**
SendCmd('Acknowledged') to <Node p>
update local info with info in <dataPacket>
do <command> using <dataPacket>

**On event: Acknowledged():**
$ack \leftarrow 1$;

**SendCmd(command, dataPacket);**

$ack \leftarrow 0$ ; //assume only one TimeOut is active at any point of time
send <command> and <dataPacket> to <Node q>
TimeOut();

**On event: TimeOut();**

WHILE NOT time_out,
//wait for ack from node to which it sent cmd
IF $ack = 0$,
$mis\sin g\_list \leftarrow mis\sin g\_list \cup \{q\}$
SkipConstraint($j$);

# Chapter 5. Application case study of DBE

## 5.1 Introduction

To demonstrate the distributed bias estimation (DBE) algorithm proposed in Chapter 4, the experimental-scale plant described in Chapter 3 is again used to conduct a case study. In this chapter however, Sensors 1, 3 and 4 are deliberately miscalibrated in order to study bias estimation. The performance of the bias estimators are studied using data contaminated with outliers (Section 5.2). The experiments were performed with 25% of the sensor measurements coming from distributions with standard deviation 3 times greater than the rest. This is a commonly used distribution to study outliers that are close to good, normal data such that they cannot be separated easily [14]. This distribution also approximates Student's $t$ with 3 degrees of freedom for the estimation problem [13, 15-16]. The results shows that the GT based estimator with sample size of 43 achieves the same estimation variance as the IQR+LS estimator with sample size of 50, showing that GT based estimation is more efficient in situation where the outliers are difficult to distinguish from the good,

normal data.

## 5.2 Performance of the bias estimators

A time window of 50 sampling instances is defined for the distributed BE. This means that the distributed BE is applied at every 50th sampling instance. The variance of the collected data set, consisting of 10,000 sets of 50 samples each, is computed to give

$$\begin{aligned}\Lambda &= diag([\Lambda_1,\ldots,\Lambda_5]) \\ &= diag([0.3687 \quad 0.1151 \quad 0.8403 \quad 1.1995 \quad 0.0326]\times 10^{-2}).\end{aligned}$$

The GT distribution parameters $\sigma_i$, $p_i$ and $q_i$ are chosen as

$\sigma_1 = 8.59\times10^{-2}$ , $\sigma_2 = 4.80\times10^{-2}$ , $\sigma_3 = 1.30\times10^{-1}$ , $\sigma_4 = 1.55\times10^{-1}$ , $\sigma_5 = 2.55\times10^{-2}$,

$p_i = 2$, $q_i = 2.31$; $i = 1, \ldots, 5$.

These values are chosen because it is known that the GT probability density function $f(z;\sigma,p,q)\big|_{p=2,q\to\infty}$ gives the probability density of a normal distribution with expectation zero and variance $\sigma^2/2$ [10]. We therefore set $p_i = 2$, $\sigma_i = \sqrt{2\Lambda_i}$ and choose $q_i = 2.31$ such that when there is no outlier contamination, the estimator produces the same estimation variance as IQR+LS. Notice that the variances of GT and IQR+LS in Table 5.1 are approximately equal.

For each estimator, the mean, variance and sample size serve as three important performance indicators. While the mean checks whether the estimator is biased, the variance and sample size quantify the precision and efficiency respectively. The theoretical variances of bias estimates by the five estimators are calculated using (4.5) and (4.15) and the results are shown in Table 5.1. A sample calculation for the variance entries of the IQR+LS and GT estimators in Table 5.2 is given in Appendix 5.

To investigate the behaviour of the estimators in the presence of outliers, outliers are introduced into the off-line data as follows. Take for example Sensor 1. First, the mean value of $y_1$ is computed. Next, 25% of randomly selected $y_1$ have their values changed such that their distances from the mean value are increased by 3 times. The same thing is done for Sensors 2-5. Thus, an expected 25% of the data points come from noise distributions with standard deviation 3 times greater than the rest. They act as probable outliers that are close to the good, normal data [14]. Bias estimation is then performed and the results are shown in Table 5.2.

Table 5.3 is the same as Table 5.2 except that 10% instead of 25% of the data have their distances from the mean value increased, by 10 times instead of 3 times. Thus an expected 10% of the data points come from distributions with standard deviation 10 times greater than the rest. They are probable outliers that are far away from the good, normal data. Bias estimation is again performed and the results are shown in Table 5.3.

Table 5.1: Bias estimation results for data without outliers

| Estimator | % average difference between estimated & true bias $(e^* - e)$ | | | Variance of estimated bias ($e^*$) $(\times 10^{-3})$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Node 1 | Node 3 | Node 4 | Node 1 | | Node 3 | | Node 4 | |
| | | | | Theory | Expt | Theory | Expt | Theory | Expt |
| LS | 0.0062 | 0.0996 | 0.2240 | 0.0967 | 0.0962 | 0.1971 | 0.1992 | 0.2461 | 0.2434 |
| IQR+LS | 0.0079 | 0.0889 | 0.2217 | 0.1033 | 0.1028 | 0.2105 | 0.2177 | 0.2628 | 0.2656 |
| GT | 0.0026 | 0.0928 | 0.2234 | 0.1033 | 0.1020 | 0.2105 | 0.2130 | 0.2628 | 0.2608 |

Table 5.2: Bias estimation results for data with 25% outliers 3 times larger than original data

| Estimator | % average difference between estimated & true bias $(e^* - e)$ | | | Variance of estimated bias ($e^*$) $(\times 10^{-3})$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Node 1 | Node 3 | Node 4 | Node 1 | | Node 3 | | Node 4 | |
| | | | | Theory | Expt | Theory | Expt | Theory | Expt |
| LS | 0.0140 | 0.1073 | 0.2387 | 0.2902 | 0.2712 | 0.5913 | 0.5535 | 0.7383 | 0.6691 |
| IQR+LS | -0.0054 | 0.1022 | 0.2307 | 0.1959 | 0.1789 | 0.3992 | 0.3719 | 0.4984 | 0.4598 |
| GT (50 samples) | -0.0042 | 0.0988 | 0.2287 | 0.1690 | 0.1590 | 0.3443 | 0.3308 | 0.4299 | 0.4072 |
| GT (43 samples) | 0.0098 | 0.1192 | 0.2249 | 0.1965 | 0.1814 | 0.4011 | 0.3879 | 0.4990 | 0.4826 |

Table 5.3: Bias estimation results for data with 10% outliers 10 times larger than original data

| Estimator | % average difference between estimated & true bias $(e^* - e)$ | | | Variance of estimated bias ($e^*$) $(\times 10^{-3})$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Node 1 | Node 3 | Node 4 | Node 1 | | Node 3 | | Node 4 | |
| | | | | Theory | Expt | Theory | Expt | Theory | Expt |
| LS | -0.0287 | 0.1039 | 0.2306 | 1.0544 | 1.0166 | 2.1484 | 2.0757 | 2.6825 | 2.5179 |
| IQR+LS | 0.0072 | 0.1027 | 0.2244 | 0.1194 | 0.1183 | 0.2432 | 0.2531 | 0.3037 | 0.3099 |
| GT | 0.0022 | 0.0932 | 0.2261 | 0.1256 | 0.1219 | 0.2558 | 0.2576 | 0.3194 | 0.3152 |

In Table 5.1, the variances of the LS estimator are the lowest. LS bias estimation formulations assume that process data follows normal (Gaussian) distribution. However, the presence of outliers can spoil the statistical analysis completely. Hence it is not wise to use a least-square algorithm without any built-in check [11,12]. Note that the variances of the LS estimators are the largest in Tables 5.2 and 5.3 because of outliers.

In Table 5.2, the variances of the GT based estimator for a sample size of 50 are the lowest. The reduction in estimation variances by GT translated to fewer samples needed to achieve specified estimation precision. Table 5.2 also shows that the GT based estimator with a sample size of 43 achieves the same variances as the IQR+LS estimator with a sample size of 50. Although by rejecting some outliers, IQR+LS achieves estimation variances substantially lower than those of simple LS, its estimation variances are not as low as those of GT. This can be understood by considering a simple case. Let both the IQR+LS estimate and the GT estimate be 0. Let the number of accepted data after the IQR test be $S'$. Suppose the value of one data point changed from $y_H^+$ to $y_H^-$. This data point which was previously rejected is now accepted by the IQR test, and the resultant LS estimate changes from 0 to $y_H^-/(S'+1)$. The GT estimate will not change appreciably because $y_H^- \approx y_H^+$. Hence, if there is a large probability of having data around the rejection points, the binary decision mode of the IQR rejection test increases the variance of the estimate appreciably.

In Table 5.3, where outliers are clearly far away from the good data, IQR+LS is effective in rejecting them as the variances of IQR+LS are the lowest. However, this problem is easy to deal with as the outliers are so obviously separated from the good data that they are easily identified. Finally, by the negligible differences between the

estimated and true biases in Tables 5.1, 5.2 and 5.3 (i.e. all less than 0.3% of the true bias), the estimates may be considered accurate.

Figure 5.1 gives the plot of the IF of LS, IQR+LS and GT used in Table 5.2. It is clear from (4.15) and (4.31) that variance varies with IF. The IF of LS is proportional to the distance between the observation and the origin. A distant observation, therefore, has a large contribution to the variance. The IF of GT is continuous and decreases to zero as observations tend to infinity. This means a distant outlier has zero influence on the estimate and explains the robustness exhibited by this estimator against outliers. The same thing can be said of IQR+LS whose IF jumps to zero if the outlier lies beyond rejection points.

The results in Table 5.2 also demonstrate that the window size affects the estimation variance. In rows 3 and 4 of Table 5.2, the GT estimator is applied on sample sizes of 50 and 43, respectively, where the estimate variance for the sample size of 50 is lower than that for the sample size of 43. This, in fact, has been hypothesized by the variance equation (4.15), i.e.

$$\mathrm{var}(\mu(f)) \approx \tfrac{1}{S}\int_{-\infty}^{+\infty} IF^2(\varepsilon) f(\varepsilon) d\varepsilon ,$$

which shows that the estimation variance is inversely proportional to the window size (i.e. the sample size of the estimator). Therefore, using the variance equation, one can use the sample size as a tuning parameter to achieve desired estimator precision.

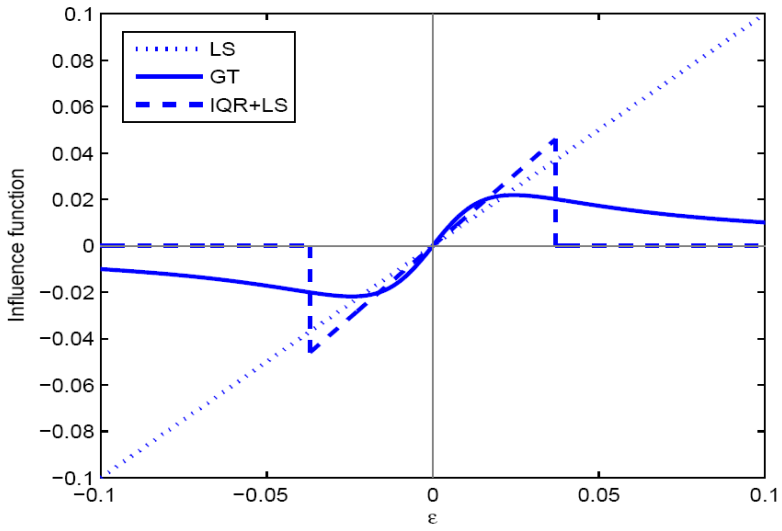



Figure 5.1. Influence functions (IF's) of the LS, GT and IQR+LS estimators

## 5.3 Conclusion

The DBE algorithm proposed in Chapter 4 has been applied to an experimental-scale chemical plant. Through the case study, the robustness of the GT-based estimator to outliers has been demonstrated. By comparing the variances of the bias estimators used for data that are contaminated with outliers, it is verified that the GT and IQR+LS estimators perform better than the LS estimator. However, the GT estimator performs better than IQR+LS for the case where outliers are difficult to distinguish from good data. Finally, by comparing variances obtained from experimental data with the theoretical variances obtained from the Influence Function (IF), it is shown that the theoretical variances provide a good indication of the actual variances.

# Appendix 5

## Sample calculation of theoretical variance

This section gives a sample calculation for the variance entries of the IQR+LS and GT estimators in Table 5.2.

For IQR+LS, $\Omega_1$ is calculated using (4.15) as

$$\Omega_1 = \mathrm{var}(\bar{y}_1) = \frac{1}{50}\int_{-\infty}^{+\infty} IF^2(\varepsilon) f(\varepsilon) d\varepsilon = 1.492\times 10^{-4},$$

where from (4.24)

$$IF(\varepsilon) = \begin{cases} 1.243\varepsilon & \text{if } -0.037 \le \varepsilon \le 0.037, \\ 0 & \text{otherwise.} \end{cases}$$

Since 75% of the observations come from a normal distribution characterized by the variance matrix $\Lambda$ and 25% of the observations come from a normal distribution characterized by the variance matrix $3^2\Lambda$, the underlying probability density function is given as

$$f(\varepsilon) = \frac{0.75}{\sqrt{2\pi\Lambda_1}}\exp\left(-\frac{\varepsilon^2}{2\Lambda_1}\right) + \frac{0.25}{3\sqrt{2\pi\Lambda_1}}\exp\left(-\frac{\varepsilon^2}{3^2\cdot 2\Lambda_1}\right).$$

The IQR test rejection points of $\varepsilon_L = -0.037$ and $\varepsilon_H = -0.037$ have been obtained using (4.21) and (4.22), respectively. The other elements of $\Omega$ can be found likewise,

giving

$$\Omega = diag\left(1.492\times10^{-4},\quad 4.670\times10^{-5},\quad 3.393\times10^{-4},\quad 4.853\times10^{-4},\quad 1.320\times10^{-5}\right)$$

Using (4.5),

$$\Phi = \begin{bmatrix} 0.1959 & 0.0968 & 0.0839 \\ 0.0968 & 0.3992 & 0.0906 \\ 0.0839 & 0.0906 & 0.4984 \end{bmatrix} \times 10^{-3}.$$

The diagonal elements are listed in the "Theory" columns in Row 2 of Table 5.2.

For GT, $\text{var}(\bar{y}_1)$ is calculated using (4.15) as

$$\Omega_1 = \text{var}(\bar{y}_1) = \frac{1}{50}\int_{-\infty}^{+\infty} IF^2(\varepsilon) f(\varepsilon) d\varepsilon = 1.287\times10^{-4},$$

where from (4.27)

$$IF(\varepsilon) = \frac{(2q+1)\varepsilon}{q\sigma^2+\varepsilon^2}\left(\int_{-\infty}^{+\infty}\frac{(2q+1)(q\sigma^2-\varepsilon^2)}{(q\sigma^2+\varepsilon^2)^2} f(\varepsilon) d\varepsilon\right)^{-1}\Bigg|_{\sigma=8.59\times10^{-2},\, p=2, q=2.31}.$$

The other elements of $\Omega$ can be found likewise, giving

$$\Omega = diag\left(1.287\times10^{-4},\quad 4.030\times10^{-5},\quad 2.927\times10^{-4},\quad 4.186\times10^{-4},\quad 1.140\times10^{-5}\right).$$

Using (4.5),

$$\Phi = \begin{bmatrix} 0.1690 & 0.0968 & 0.0839 \\ 0.0968 & 0.3443 & 0.0906 \\ 0.0839 & 0.0906 & 0.4299 \end{bmatrix} \times 10^{-3}.$$

The diagonal elements are listed in the “Theory” columns in Row 3 of Table 5.2.

# Chapter 6. Conclusion & future work

The distributed data reconciliation (DDR) is derived, and an implementation algorithm is developed and evaluated through simulation and case study examples. DDR enables a group of intelligent sensors to perform DR in-network. DDR is derived through equivalence transformation from the centralized DR, such that it is mathematically equivalent to the centralized DR. Reconciled estimates produced by DDR are therefore identical to those produced by centralized DR.

By applying DDR, each sensor node is made aware of itself and its neighbours and can respond to abnormal situation such as a missing/failed neighbouring node. The dependence on a central processing node to perform DR is eliminated, making DDR robust to the failure of the central processing node. This is in contrast to the conventional centralized scheme, where the availability of the central processing node is critical to ensure the viability of the DR processing. Application of DDR in an experimental-scale chemical plant demonstrates its usefulness in maintaining operation despite node failure.

DDR in its current form lays the groundwork for further exploration in distributing processing of sensor measurements. The distributed bias estimation (DBE) is next proposed, building upon the foundation provided by DDR. The distributed bias

estimation (DBE) is derived to enable sensor nodes to perform bias estimation in-network, and an implementation algorithm is developed. Similar to DDR, DBE inherits the property of robustness against node failures.

The performance of the generalized T (GT), inter-quartile range test cum least-square (IQR+LS) and least-square (LS) estimators are also analyzed through both theoretical tools and experiments. The Influence Function (IF) is used to theoretically quantify the estimator variance. Through comparison with the actual variances of the experiment results, the theoretical variance is shown to provide a good indication of the actual variance. As the theoretical variance is a function of the sample size used in the estimation, it can be used as a tool to select the most suitable estimator for a specified sample size, or to decide the best sample size for a particular estimator to yield the desired performance criterion (i.e. the variance). In the case study of Chapter 5, it is found that the GT estimator is the most efficient when it is required that the DBE be performed more often, i.e. with smaller sample size, in the case where the 25% outliers are approximately three standard deviations away from the mean.

With the estimator variance as a comparison tool, it is found that GT and IQR+LS behave better, i.e. have smaller variances, than LS when the data are corrupted with outliers. A qualitative explanation to this is given by the Influence Function (IF) plots. The IF plots show that GT and IQR+LS suppress large minority noises that are considered outliers, by assigning decreasing and zero weights to such data, respectively. LS, on the other hand, assigns weights that are proportional to the data magnitude, resulting in the distortion of its estimates by even very few outliers. Between GT and IQR+LS, the performance of GT is less sensitive to the magnitude of the outliers in cases where it is difficult to distinguish the outliers from the good data, while IQR+LS becomes slightly less efficient in such cases as the outliers lie

around its rejection boundaries.

# Future work

More extensive application of the proposed DDR and DBE in sensor network deployments is desirable, to identify practical issues that typifies sensor network deployments. With such issues identified, anticipative strategies can then be incorporated into the DDR and DBE to make it more robust in real deployment. A practical issue of interest is latency and synchronization of nodes in the network. The proposed approaches are based on propagation from one node to its neighbour until all nodes in the network have been involved. In a very large-scale network, it is possible that parts of the network form sub-networks with minimal but non-negligible relationships among one another. It is therefore possible to improve latency in such a large-scale network by introducing more parallelism into the coordination protocols. An example is to consider cluster-based network architecture, where each cluster comprises nodes that are highly correlated with one another, while between the nodes in one cluster and another, there is some but limited correlation. Each cluster could then be running a local DDR/DBE in parallel, after which the clusters collaborate to solve the global DDR/DBE. The current DDR and DBE algorithms have the potential and flexibility for such extension.

Another possible future study involves integrating the DDR/DBE algorithms with diverse sensor network routing protocols. This is also highly related to communication links between the nodes. For example, if it is of interest to conserve communication energy, the routing protocol that best achieves this under the current

communication link configuration can be incorporated with the DDR/DBE algorithms.

The distributed DR algorithm applies to any number of nodes *N*, in the sense that it is not affected in terms of the correctness of the solutions. This follows from the equivalence between the batch and the sequential algorithm, which holds for any size of the matrix *A*. Scalability issues would present themselves in practical implementation considerations. An example would be the time needed to obtain final solutions. As the network size grows, a round of DR will take longer to solve as messages are passed among larger number of nodes. In the conventional "batch" DR, larger network size means larger matrix need to be inverted, where the increase in the computation complexity is in $O(N^3)$ (*N* is the number of nodes in the network).

A possible strategy to deal with this issue is by considering a modular implementation of DDR, where instead of processing every single row of the constraint matrix A sequentially, groups are formed by several rows of *A* each. Each group of rows can be processed at the same time to produce the group estimates, after which sequential processing between the groups of rows will produce the global estimates.

This strategy is equivalent to grouped the nodes into clusters, such that nodes within a cluster are more highly related to one another than to nodes in any other clusters. Such grouping translates to achieving near block-diagonality of the constraint matrix *A*, and then defining the blocks in the diagonal as the group of rows or clusters of nodes as discussed above. Nodes within two clusters that do not share common nodes could perform DDR in parallel rather than in sequence, after which the intermediate solutions by the two clusters can be reconciled through a common neighbouring cluster. The parallelism would reduce solution time, and there is potential to add more

parallelism, by introducing groupings of clusters, for example.

Certain features might be of particular importance in a measured system. For example, a sensor network could be deployed to measure the probability of the occurrence of a critical event, where the probability is computed from the measurements of several key sensors. In such a case, the importance of the key sensors can be reflected as certain weight parameters in the formulation of the DR/BE problem. In its current form, the DR/BE weigh the sensors based on only the accuracy of the measurements, e.g. the variance of the measurements.

Correlation of the measurement noises could be too large to omit in certain measurement systems. In the DDR, correlation can be taken into account by the use of non-diagonal measurement covariance matrix $\Psi$ (see Example in Appendix 2C). Several techniques exist in the DR literature to estimate the covariance matrix $\Psi$ [35,36]. It would be of interest to examine such techniques under the distributed implementation framework. Furthermore, using correlated data might affect the result of further analysis on the data. Data decorrelation techniques are also available in the literature [37-39], and in the DR literature, such techniques have been used in conjunction with gross error tests [39]. It could be of interest to investigate how such decorrelation techniques can be incorporated into the DDR.

DDR and DBE have been formulated for the linear, steady-state case. The current state of the art for dynamic DR/BE is based on Kalman Filter for the linear dynamic system with Gaussian assumption on measurement noise [32-34]. One could consider extending the DR/BE with GT noise distribution algorithm in the thesis to the Kalman filter.

# Bibliography

[1] M. A. Paskin and C. E. Guestrin, "Robust probabilistic inference in distributed systems", in *Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, Banff, Canada, July 2004, pp. 436-445.

[2] A. Bharathidasan and V. A. S. Ponduru, "Sensor networks: an overview", in *The 23rd Conference of the IEEE Communications Society (Infocom 2004)*, Hong Kong, March 2004.

[3] L. Balzano, "Addressing fault and calibration in wireless sensor networks", *M.S. Thesis*, Dept. Elect. Eng., Univ. of California, Los Angeles, 2007.

[4] J.A. Romagnoli and M.C. Sanchez, *Data Processing and Reconciliation for Chemical Process Operations*, Academic Press, 2000.

[5] D. Tulone and M. Srivastava, "Inspect: a general framework for on-line detection and diagnosis of sensor faults", in *Proceedings of the 2nd International Conference on Internet Technologies & Applications (ITA)*, Wrexham, North Wales, UK, September 2007.

[6] L. Balzano and R. Nowak, "Blind calibration of sensor networks", in *Proceedings of the 6$^{th}$ International Conference on Information Processing in Sensor Networks (IPSN)*, April 2007, pp. 79-88.

[7] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. “A collaborative approach to in-place sensor calibration”, *Lecture Notes in Computer Science*, Vol. 2634, pp. 301–316, 2003.

[8] Joe, Y.Y., Wang, D., Tay, A., Ho, W.K., Ching, C.B., and Romagnoli, J.A. “A Robust Strategy for Joint Data Reconciliation and Parameter Estimation”, *Proceedings of the 14th European Symposium on Computer-Aided Process Engineering,* CACE, Elsevier, Lisbon, Portugal, 2004.

[9] C. Hansen, J. B. McDonald and W. K. Newey. “Instrumental variable estimation with flexible distributions”, *Journal of Economics and Business Statistics*, 2007.

[10] J. B. McDonald, W. K. Newey, “Partially adaptive estimation of regression models via the Generalized T distribution”, *Econometric Theory*, Vol. 4, pp. 428-457, 1998.

[11] D. Wang and J. A. Romagnoli. “A framework for robust data reconciliation based on a generalized objective function”, *Industrial & Chemical Engineering Research*, Vol. 42, pp. 3075-3084, 2003.

[12] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw and W. A. Stahel. *Robust Statistics – the Approach based on Influence Functions*, John Wiley & Sons, 1986.

[13] J. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.

[14] D. F. Andrews, P. J. Bickel, F. R. Hampel, P. J. Huber, W. H. Rogers and J. W. Tukey, *Robust Estimates of Location: Survey and Advances*, Princeton University Press, 1972.

[15] F. R. Hampel, “The breakdown points of the mean combined with some rejection rules”, *Technometrics*, Vol. 27, No. 2, 1985.

[16] W. N. Venables, B. D. Ripley, *Modern Applied Statistics with S*, Springer-Verlag, New York, 2002.

[17] S. Narasimhan, M. Jordache, *Data Reconciliation and Gross Error Detection: An Intelligent Use of Process Data*, Gulf Publishing Co., 1999.

[18] Crowe, C. M., "Data reconciliation – Progress and challenges", *Journal of Process Control*, Vol. 6, No. 2-3, pp. 89-98, 1996.

[19] F. Madron, *Process Plant Performance: Measurement and Data Processing for Optimization and Retrofits*, Ellis Horwood, Chichester, England, 1992.

[20] R. S. H. Mah, "Chemical Process Structures and Information Flows", *Chemical Engineering Series*, Butterworth, Boston, 1990.

[21] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors", in *WSNA '03: Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, 2003.

[22] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks", in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pp. 59–67, 2002.

[23] J. Feng, S. Megerian, and M. Potkonjak, "Model-based calibration for sensor networks", *Sensors*, pp. 737 – 742, 2003.

[24] A. Ihler, J. Fisher, R. Moses, and A. Willsky. "Nonparametric belief propagation for self-calibration in sensor networks". In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

[25] G. Tolle and J. P. et al, "A macroscope in the redwoods", in *Proceedings of Sensys*, 2005.

[26] N. Ramanathan and L. B. et al, "Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks". *Technical Report CENS TR 62*, Center for Embedded Networked Sensing, 2006.

[27] P. Ji and M. Szczodrak. "A multivariate model for data cleansing in sensor networks", in *ACITA 2008*: *2nd Annual Conference of the International Technology Alliance*, London, 2008.

[28] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, J. Widom, "A Pipelined Framework for Online Cleaning of Sensor Data Streams", in *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, pp.140, April 03-07, 2006.

[29] S. Mukhopadhyay, D. Panigrahi, S. Dey, "Data aware, low cost error correction for wireless sensor network", in *Wireless Communication and Networking Conference*, 2004.

[30] Y. L. Tan, V. Sehgal, H. H. Shahri, "SensoClean: Handling noisy and incomplete data in sensor networks using modeling", *Technical Report*, University of Maryland, 2005.

[31] A. Speranzon, C. Fishione, K. H. Johansson and A. Sangiovanni-Vincentelli, "A distributed minimum variance estimator for sensor networks", in *IEEE Journal on Selected Areas in Communications*, Vol. 26, No. 4, pp. 609-621, May 2008.

[32] D. P. Spanos, R. Olfati-Saber, R. M. Murray, "Distributed Kalman Filtering in Sensor Networks with Quantifiable Performance", in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, April 24-27, 2005.

[33] H. F. Durrant-Whyte, B. Y. S. Rao, and H. Hu, "Toward a fully decentralized architecture for multi-sensor data fusion", in *Proceedings of the IEEE*

*International Conference on Robotics and Automation*, Vol. 2, pp. 1331–1336, 1990.

[34] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks", in *Proceedings of the 46th Conference on Decision and Control (CDC)*, pp. 5492–5498, New Orleans, LA, USA, Dec 2007.

[35] G. A. Almasy and R. S. H. Mah, "Estimation of measurement error variances from process data", in *Industrial and Engineering Chemistry Process Design and Development*, Vol. 23, pp. 779-784, 1984.

[36] J. Chen, A. Bandoni and J. A. Romagnoli, "Robust estimation of measurement error variance/covariance from process sampling data", in *Computers and Chemical Engineering*, Vol. 21, pp. 593-600, 1997.

[37] L. H. Chiang, E. L. Russell and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*, Springer-Verlag, London, 2001.

[38] T. Kourti and J. F. MacGregor, "Process analysis, monitoring and diagnosis, using multivariate projection methods", in *Chemometrics and Intelligent Laboratory Systems*, Vol. 28, pp. 3, 1995.

[39] H. Tong and C. M. Crowe, "Detection of gross errors in data reconciliation by principal component analysis", in *American Institute of Chemical Engineers (AIChE) Journal*, Vol. 41, pp. 1712-1722, 1995.

# Author's Publications

Joe, Y.Y., Ling, K.V., Ho, W.K., Lim, K.W. (2009). **Distributed Data Reconciliation for Sensor Network**, *submitted to IEEE Transactions on Industrial Informatics*.

Joe, Y.Y., Ding, Z.Q., Zhang, J.B., Ling, K.V., Ho, W.K., Romagnoli, J.A., Lim, K.W. (2006). **Clustering Intelligent Sensor Nodes for Distributed Fault Detection and Diagnosis**, 4rd International IEEE Conference on Industrial Informatics, Singapore.

Joe, Y.Y., Ding, Z.Q., Ling, K.V., Romagnoli, J.A. (2005). **An Intelligent Sensor Network for Distributed Data Rectification and Process Monitoring**, 3rd International IEEE Conference on Industrial Informatics, Perth, Australia.

Joe, Y.Y., Xu, H., Dong, Z.Y., Ng, H.H., and Tay, A. (2004). **Searching Oligo Sets of Human Chromosome 12 using Evolutionary Strategies**, International Journal of Systems Sciences, Vol.35, No.13-14, Taylor and Francis Ltd, London.

Joe, Y.Y., Wang, D., Tay, A., Ho, W.K., Ching, C.B., and Romagnoli, J. (2004). **A Robust Strategy for Joint Data Reconciliation and Parameter Estimation.** Proceedings of the 14th European Symposium on Computer-Aided Process Engineering. CACE, Elsevier, Lisbon, Portugal.